

Commande des procédés industriels avec des logiciels libres

Simone Mannori , Ramine Nikoukhah , Serge Steer
INRIA-Rocquencourt, Domaine de Voluceau,
78153 Le Chesnay Cedex, France
simone.mannori@inria.fr

Abstract— Il existe aujourd’hui des logiciels libres pour la conception, le développement, la génération automatique de code et le test de régulateurs suffisamment matures et stables pour être des alternatives aux solutions propriétaires dans le cadre de la recherche, de l’éducation et des applications industrielles. Nous présentons, sur un exemple réel, une chaîne complète de développement depuis les outils d’analyse et de conception du régulateur jusqu’à la génération automatique du code pour les contrôleurs embarqués et son interface utilisateur. Cette chaîne est basée sur les logiciels libres Scicos, Scilab, Comedi, RTAI et Linux.

Mots-clés: Commande temps réel; simulation des systèmes dynamiques; logiciels de simulation; systèmes hybrides; Scilab; Scicos; RTAI; Comedi

I. INTRODUCTION

Nous présentons, sur un exemple réel, une plate-forme de développement complète qui permet d’effectuer toutes les étapes nécessaires à la conception et à la validation d’un régulateur pour un procédé physique:

- construire un modèle mathématique du procédé,
- valider le modèle par simulation et par acquisition de mesures sur le procédé,
- concevoir le régulateur avec les outils de l’automatique,
- simuler le système formé du modèle du procédé et de son régulateur et d’optimiser les paramètres du régulateur,
- simuler le système formé du régulateur numérique et du procédé physique (simulation “hardware-in-the-loop”),
- générer le code qui pourra être embarqué sur un calculateur temps réel.

Dans ce travail nous avons cherché à limiter les coûts en utilisant des composants standards et peu onéreux. Cela est fait en particulier en utilisant les logiciels libre Scilab et Scicos.

II. CONSTRUCTION DU SYSTÈME PHYSIQUE À CONTRÔLER

La façon la plus rapide pour disposer d’un système physique de démonstration est de l’acheter tout prêt. De

nombreuses sociétés proposent de tels produits [1], [2]. Le problème est le coût, souvent au delà du budget disponible. Nous avons choisi le système test bien connu de la bille roulant sur un bras pivotant (Fig. 1 (a)) à cause de son instabilité intrinsèque et des nonlinéarités et de la possibilité de construire le système en utilisant des composants disponibles peu chers. Pour la plupart des composants mécaniques nous avons utilisé les composant “LEGO Technique/Mindstorm” [3], [4] (moins de 200 euros pour un kit complet).

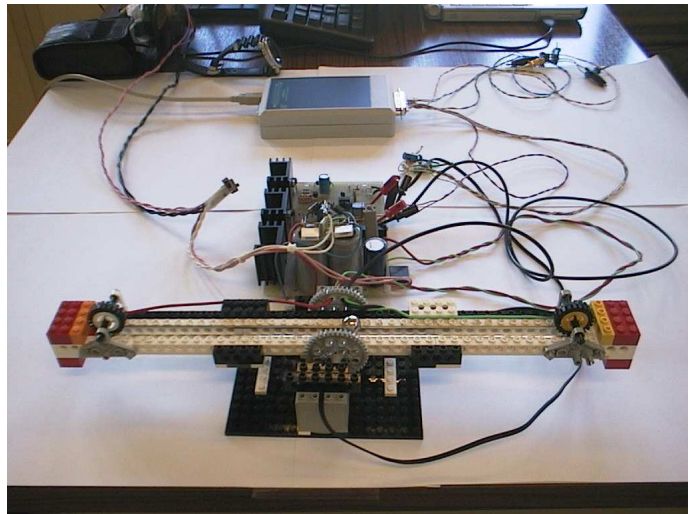


Fig. 1. Système Complet. (a) le bras et la bille; (b) Les amplificateurs et l’alimentation électrique; (c) la carte analogique digitale USB-DUX.

Nous avons réalisé l’amplificateur de mesure, l’amplificateur de puissance et l’alimentation sur une carte imprimée expérimentale. La bille est en acier inoxydable ; elle roule sur deux rails récupérés sur un train électrique et collés sur les briques LEGO (Fig. 1(a)).

A. Mesure de la position de la bille

La position de la bille est mesurée en utilisant l’un des rails comme une source de tension variable (Fig. 2). Une source de tension flottante de 5V fournit un courant constant

de $700mA$ dans le rail de référence. La consommation de ce système est élevée car il dissipe beaucoup d'énergie dans le circuit d'alimentation qui fournit la source de tension stabilisée de $5V$ et dans la résistance de puissance externe (6.8Ω , $5Watts$). Pour la prochaine version, nous pensons utiliser une alimentation à commutation de $100mV$. La bille métallique joue le rôle d'un contact glissant ; le second rail ferme le circuit ce qui envoie un signal ($0-40mV$, $0-30cm$) à l'amplificateur de mesure.

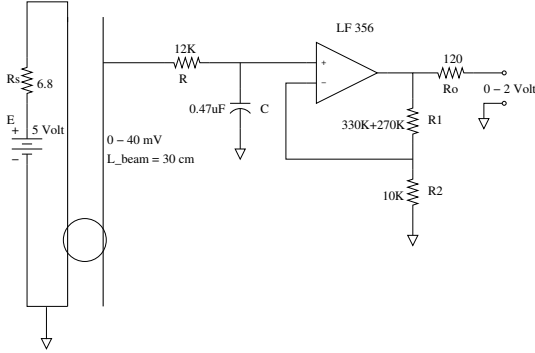


Fig. 2. Amplificateur de mesure.

La résistance de contact entre la bille et le rail est très bruitée. L'amplificateur de mesure (Fig. 2) a une forte impédance d'entrée et utilise un filtre passe-bas RC qui coupe une partie du bruit de ce capteur. Nous avons fixé deux constantes de temps ($5ms$ and $0.5ms$) compatibles avec la dynamique prévue pour le système boucle fermée. Grâce à l'impédance importante de l'amplificateur opérationnel JFET LF356, si la bille perd le contact électrique, le filtre de mesure fonctionne comme un bloqueur ("Sample-and-Hold"), la capacité C "mémorisant" la dernière valeur correcte de la tension (donc de la position).

B. Actionneur

Pour ce qui est de l'actionneur nous utilisons le modèle standard ("71427") de moteur du kit LEGO Mindstorm. Les spécifications de ce moteur sont fournies [5]. L'inductance de l'armature du moteur ($L_A = 2.07mH$) n'est pas présente dans le modèle car elle introduit une constante de temps négligeable.

$$\tau_e = L_A/R_A = 0.83ms$$

Du fait du rendement électromécanique limité du moteur, les valeurs K_T (couple constant) et K_V (vitesse constante) sont différentes (elles sont égales dans le cas d'un moteur parfait c.à.d. ayant un rendement de conversion électromécanique de 100%). Ces constantes diffèrent aussi

car le moteur LEGO contient un réducteur de vitesse qui réduit l'efficacité mais augmente le couple et améliore sa linéarité pour les faibles vitesses. Les inerties du moteur et du réducteur de vitesse sont reportées dans le moment d'inertie du modèle du bras.

Dans le modèle, nous négligeons toutes les autres sources de frottement.

Le moteur est suffisamment puissant pour déplacer le bras et la bille directement, mais nous avons utilisé un réducteur de vitesse (roues dentées) externe pour faire une meilleure utilisation de la plage de vitesse du moteur.

C. Amplificateur de puissance

La sortie de l'amplificateur opérationnel est à nouveau amplifiée par une paire de transistors Darlington. Cet amplificateur de puissance linéaire a une bande passante en fréquence ($10kHz$), une dynamique de tension ($\pm 15V$) et d'intensité ($\pm 1.5A$) largement suffisantes pour piloter le moteur et sa charge. La tension est limitée à $\pm 9V$ pour éviter une surcharge du moteur (qui pourrait l'endommager). Nous avons choisi un gain en voltage unitaire pour l'amplificateur de puissance car nous avons trouvé qu'une dynamique de $\pm 4V$ pour la tension de sortie de l'USB-DUX est suffisante pour piloter le moteur (l'amplificateur de sortie fonctionne uniquement comme un amplificateur de courant).

D. Entrée "encoder"

Il est possible d'utiliser le compteur bi-directionnel de la carte d'acquisition (USB-DUX) pour connecter un codeur permettant de mesurer l'angle du bras. Nous n'avons pas utilisé cette entrée dans l'expérience car elle n'est pas nécessaire pour rendre le système observable. Nous avons ainsi évité d'utiliser un capteur supplémentaire pour mesurer la position angulaire de la barre. Un observateur numérique permet d'obtenir une estimation de l'ensemble des variables d'état du système à partir de la mesure de la position de la bille.

E. Carte d'acquisition de donnée

Nous avons utilisé l'interface USB-DUX [6]. Nous avons choisi ce type de carte car il a suffisamment de ressources pour acquérir les données et piloter notre système. La carte USB-DUX utilise une connection USB qui est disponible sur tous les PC récents. L'interface USB-DUX utilise le pilote contenu dans la bibliothèque Comedi [7] qui permet d'utiliser des cartes différentes avec une interface (API) commune. Dans l'esprit du logiciel libre et ouvert, la carte USB-DUX est complètement documentée, les schémas

électroniques ainsi que le source du micro-contrôleur sont publiés. Enfin le prix est raisonnable (250 euros). La seule vraie limite de cette carte est le bus USB ; malheureusement il n'est pas encore possible de faire fonctionner ce bus en temps réel car pour faire cela il est nécessaire de réécrire complètement la pile USB sous Linux [8]. Avec les noyaux Linux récents, ce n'est pas une limitation très pénalisante pour des applications ayant une période d'échantillonnage de 10ms ou plus. S'il était nécessaire de garantir une réelle exécution temps réel, il serait alors nécessaire d'utiliser une carte d'entrées/sorties ISA/PCI/PC-CARD [9] avec RTAI.

III. PRÉSENTATION DE SCILAB/SCICOS ET LEUR UTILISATION

Scilab [10], [11], [12] est un logiciel de calcul numérique qui fournit un environnement puissant pour le calcul scientifique. Depuis 1994, ce logiciel est distribué sur Internet, avec ses sources, sous une licence libre. Il est utilisé pour l'éducation et l'industrie dans le monde entier. Ce logiciel inclut des centaines de fonctions mathématiques et permet d'ajouter interactivement des programmes écrits dans différents langages de programmation (FORTRAN, C, C++, Scilab, JAVA). Il comprend un langage de programmation interprété de haut niveau, des structures de données complexes (listes, polynômes, fractions rationnelles, systèmes dynamiques linéaires,...). Scilab comprend une boîte à outils complète pour la modélisation, la simulation et la conception de régulateurs hybrides.

Scicos [13], [14] est un éditeur graphique et un simulateur de systèmes dynamiques inclus dans Scilab. L'éditeur graphique permet de réaliser des schémas bloc-diagrammes représentant des systèmes dynamiques hybrides, de les simuler et de les compiler sous forme d'un code exécutable. Scicos est utilisé pour des applications de traitement du signal, de contrôle, pour modéliser et simuler des files d'attente, pour étudier des systèmes physiques ou biologiques,...

Dans le cadre de ce projet, nous avons utilisé les outils logiciels Scilab et Scicos. Dans un premier temps nous avons écrit le modèle mathématique non linéaire du procédé et en avons déduit un modèle linéaire autour du point de fonctionnement correspondant à la position horizontale du bras et une bille à l'arrêt au milieu du bras que nous avons programmé dans Scilab. Ce modèle est alors discrétisé par la méthode de "discrétisation exacte" en utilisant la fonction `dscr` de Scilab.

Ce système linéaire discret étant instable, nous avons construit un régulateur formé d'un contrôleur à base d'observateur pour le stabiliser. Pour la conception du contrôleur de retour d'état, ainsi que pour celle de l'observateur, l'approche LQ (linéaire quadratique) est

utilisée. Un intégrateur discret est aussi introduit dans le contrôleur pour rejeter tout biais constant.

La conception du régulateur à base d'observateur est effectuée dans Scilab en utilisant les fonctions de la boîte à outils dédiée à l'automatique. Nous avons utilisé en particulier les fonctions `lqr` (retour d'état LQ) et `lqe` (filtre de Kalman). Le programme complet de la conception du régulateur consiste d'un seul script Scilab nommé `bb_a.sce`.

Une fois le régulateur conçu, il est validé par simulation sur le modèle non linéaire dans l'environnement Scicos; voir Fig. 3. Le script `bb_a.sce` est exécuté par le contexte de ce diagramme.

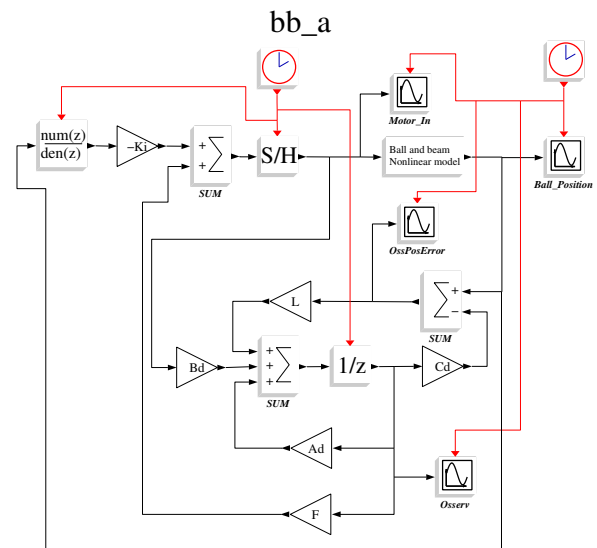


Fig. 3. Diagramme Scicos `bb_a.cos` contenant le modèle non linéaire et le régulateur linéaire discret.

Enfin, Scicos est utilisé pour faire de la simulation "hardware-in-the-loop" et pour la génération du code temps réel.

IV. VALIDATION DU RÉGULATEUR APPLIQUÉ AU PROCÉDÉ PHYSIQUE (SIMULATION "HARDWARE-IN-THE-LOOP")

Il est possible d'utiliser Scicos en ligne pour commander des procédés physiques. La première étape est de créer les blocs entrées-sorties pour interfacer Scicos avec une carte d'acquisition. À partir des documentations Scicos [12], [13] et Comedi [7], nous avons facilement développé les fonctions d'interfaçage et de calcul des nouveaux blocs d'entrées-sorties Scicos pour toutes les cartes supportées par Comedi.

La simulation Scicos peut être réalisée en “temps-réel” en utilisant le paramètre `Realtime scaling`. En fixant ce paramètre à 1, l’unité de temps Scicos correspond alors à une seconde. Bien sûr Scicos ne peut que ralentir la simulation pour respecter le rythme imposé donc il faut que les calculs nécessaires à chaque cycle prennent moins de temps que la période d’échantillonnage (10ms). Cela, pour notre application et avec des ordinateurs PC récents, ne pose pas de problème. Le respect de la fréquence d’échantillonnage est très important car la dynamique du régulateur en dépend fortement. En particulier, un décalage de cette fréquence impliquerait un changement dans le placement des poles et des zéros du régulateur discret. Cela bien sûr affecterait la stabilité et la performance du système en boucle fermée.

Pour la validation du régulateur, à part la surveillance visuelle du procédé physique (l’emplacement de la bille), on utilise des blocs `Scope` pour visualiser les différents signaux présents dans le système. Ces signaux peuvent être enregistrés et utilisés pour optimiser les paramètres du régulateur.

V. CONSTRUCTION DU RÉGULATEUR

A. Conception du contrôleur à base d’observateur

Comme période d’échantillonnage nous avons choisi $T_s = 10ms$ ce qui correspond à la fréquence d’échantillonnage maximum garantie par la carte USB-DUX utilisée dans notre maquette.

Supposons que le modèle linéaire de notre procédé soit représenté par les matrices de représentation état A , B , C et D . Le modèle discret est alors obtenue par l’instruction Scilab

```
sys_d = dscr(sys,Ts);
```

où `sys` est l’objet Scilab correspondant au système linéaire :

```
sys = syslin('c',A,B,C,D);
```

Noter que A est une matrice 4×4 (le modèle contient quatre état ; voir l’annexe), B est 4×1 (la seule entrée étant le voltage du moteur), C est 1×4 (la seule sortie étant la position de la bille) et D est nulle.

Pour inclure l’intégrateur, nous augmentons l’espace d’état en rajoutant un état supplémentaire. Les matrices A et B de nouveau système sont alors construits de la façon suivante :

```
Adi=[sys_d.A,zeros(4,1);-sys_d.C,1];
```

```
Bdi=[sys_d.B;0];
```

Le calcul de la matrice de gain F_{lqr_i} s’effectue pour ce système en utilisant la fonction `lqr` pour un choix approprié des matrices de poids Q et R . Les matrices F et K_i qu’on voit sur le schéma Scicos de Fig. 4 sont alors obtenus par :

```
F = -F_lqr_i(1:4);
```

```
Ki = -F_lqr_i(5);
```

En ce qui concerne l’observateur, l’intégrateur n’est pas pris en compte (car son état fait partie du régulateur et donc ne nécessite pas une estimation) et donc la fonction `lqe` est appliquée au système `sys_d` avec les matrices de poids (matrices de covariance) appropriées Q_{ϵ} et R_{ϵ} .

Enfin, le régulateur qui est un contrôleur à base d’observateur (dans ce cas un régulateur LQG), est obtenu en utilisant l’estimation de l’état, fournit par l’observateur (filtre de Kalman), à la place de l’état dans le contrôleur LQ.

B. Compensation de la zone morte

Il s’avère que la dynamique du moteur LEGO chargé par la mécanique, présente une zone morte qui doit être compensée par le régulateur. Cela est fait en utilisant le bloc `Mathematical Expression` avec l’expression :

```
sign(u1)*a+u1
```

Le paramètre a , défini dans le `Context`, représente l’étendue de la zone morte du moteur. Voir l’implémentation du régulateur complet en Scicos dans Fig. 4. Noter aussi l’utilisation du bloc non linéaire `Saturation` pour limiter le voltage appliqué au moteur.

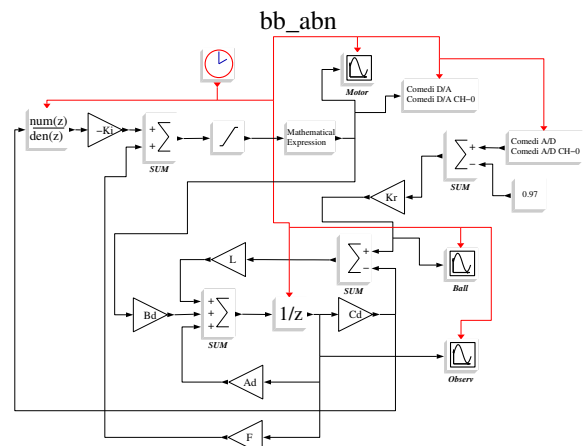


Fig. 4. Schéma Scicos pour une utilisation “hardware-in-the-loop”.

VI. GÉNÉRATION DU CODE TEMPS-RÉEL POUR LINUX RTAI

Avec Scicos, il est aussi possible de générer automatiquement du code temps-réel pour Linux RTAI. Le générateur du

code développé par Roberto Bucher [15] génère le code C et les Makefiles associés qui peuvent être exécutés sur Linux RTAI pour implémenter le régulateur. Cette solution qui est en général utilisée quand la vitesse du PC ne permet pas de suivre la cadence imposé en simulation “hardware-in-the-loop” (le code généré étant plus efficace que la simulation dans Scicos) a aussi l’avantage d’être “standalone” et donc utilisable sur un système embarqué dépourvu de Scilab.

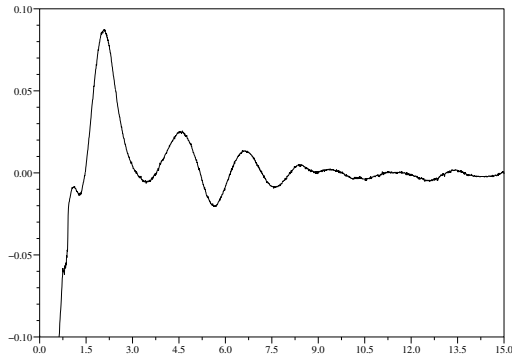


Fig. 5. Position de la bille (m) en fonction du temp (s).

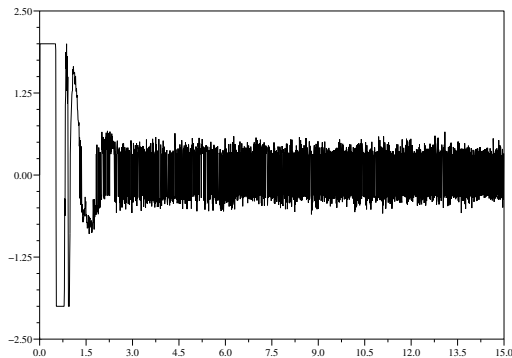


Fig. 6. Voltage du moteur (Volt) en fonction du temp (s).

VII. CONCLUSION

Dans ce papier, nous avons présenté une maquette complète et peu onéreuse qui peut être utilisée pour l’enseignement de l’automatique industrielle. Nous avons validé le fonctionnement de cette maquette, aussi bien dans le mode “hardware-in-the-loop” que par la génération du code temps réel embarqué pour Linux RTAI. Les figures 5 et 6 montrent le comportement transitoire et le régime permanent du système pendant un test où la bille est initialement placée à l’une des extrémités du bras.

REFERENCES

- [1] Quanser [Online].
Disponible: <http://www.quanser.com>.
- [2] TQ Education and Training Limited [Online].
Disponible: <http://www.tq.com/>.
- [3] LEGO [Online].
Disponible: <http://www.lego.com/>.
- [4] LEGO Mindstorm [Online].
Disponible: <http://mindstorms.lego.com/>.
- [5] LEGO motor parameters [Online].
Disponible: <http://www.philohome.com/motors/motorcomp.htm>.
- [6] USB-DUX [Online].
Disponible: <http://www.linux-usb-daq.co.uk/>.
- [7] Comedi [Online].
Disponible: <http://www.comedi.org/>.
- [8] USB-RTAI stack: works in progress [Online].
Disponible: <https://www.rtai.org/>.
- [9] PC-CARD DAS16/16-AO Measurement Computing [Online].
Disponible: www.measurementcomputing.com.
- [10] J.P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikoukhah et S. Steer, *Introduction à Scilab*, Springer, 2001.
- [11] C. Bunks, J.P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikoukhah et S. Steer, *Engineering and Scientific Computing with Scilab*, Birkhäuser, 1999.
- [12] Scilab [Online].
Disponible: <http://www.scilab.org/>.
- [13] Scicos documentation [Online].
Disponible: <http://www.scicos.org>.
- [14] S. L. Campbell, J-Ph. Chancelier et R. Nikoukhah, *Modeling and Simulation in Scilab/Scicos*, Springer, 2005.
- [15] R. Bucher [Online].
Disponible: <http://www.dti.supsi.ch/bucher/scilab-howto.pdf>

ANNEXE

MODÈLE MATHÉMATIQUE DU PROCÉDÉ PHYSIQUE

Le modèle mathématique du système mécanique constitué du bras et la bille contient trois états : la position de la bille notée x , l’angle de rotation de la bille, ϕ , et l’angle de rotation du bras, θ . Le mouvement de la bille vérifie alors

$$mg \sin(\theta) = m\ddot{x} + J_s \ddot{\phi} / r$$

où r représente le rayon de la bille (mètres), m la masse de la bille (Kg) et $J_s = 2mr^2/5$, l’inertie rotationnelle de celle-ci.

En utilisant la relation $x = \phi r$ qu’on obtient en supposant que la bille roule sans glissement sur les rails, on obtient :

$$\ddot{x} = 5g \sin(\theta) / 7.$$

L’autre équation différentielle est obtenue en faisant le bilan des couples autour de l’axe du bras :

$$T = -mg \cos(\theta)x + J_b \ddot{\theta} + mx^2 \ddot{\theta}$$

où T représente le couple exercé sur le bras par le moteur. En utilisant un modèle simplifié du moteur, en particulier en négligeant l’inductance, nous avons

$$T_M = K_T i_A$$

où T_M représente le couple moteur (Nm), K_T la constante du moteur (Nm/A), et i_A le courant d’armature du moteur (A).

En considérant le quotient de réduction due à la mécanique externe au moteur $K_C = 40/8 = \omega_M/\omega = T/T_M$ où ω_M représente la vitesse de rotation du moteur et $\omega = \dot{\theta}$, et l'équation de la dynamique du moteur

$$U_M = R_A i_A + K_V \omega_M,$$

nous obtenons

$$K_C K_T i_A = -mg \cos(\theta)x + J_b \ddot{\theta} + mx^2 \ddot{\theta}$$

où

$$i_A = \frac{U_M - K_V \omega_M}{R_A}.$$

En notant \dot{x} par v , nous obtenons le modèle complet du système :

$$\dot{\omega} = \frac{mgx}{J_b + mx^2} \cos(\theta) - \frac{K_V K_C^2 K_T}{R_A (J_b + mx^2)} \omega + \frac{K_C K_T U_M}{R_A (J_b + mx^2)}$$

$$\dot{\theta} = \omega$$

$$\dot{v} = \frac{5}{7}g \sin(\theta),$$

$$\dot{x} = v.$$

Les états stationnaires de ce système sont $(\omega_0, \theta_0, v_0, x_0) = (0, 0, 0, *)$ où $*$ indique une valeur arbitraire. Il est en effet possible de stabiliser la bille ailleurs qu'à l'origine, mais ici on ne considère que le cas $x_0 = 0$.

En linéarisant le modèle autour de l'état d'équilibre 0, nous obtenons un modèle approximatif valable autour du point d'équilibre représenté par le système LTI suivant :

$$\begin{aligned} \dot{\xi} &= A\xi + Bu \\ y &= C\xi \end{aligned}$$

où

$$\xi = \begin{pmatrix} \omega \\ \theta \\ v \\ x \end{pmatrix}, A = \begin{pmatrix} -\frac{K_V K_C^2 K_T}{R_A J_b} & 0 & 0 & \frac{mg}{J_b} \\ 1 & 0 & 0 & 0 \\ 0 & \frac{5}{7}g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} \frac{K_C K_T}{R_A J_b} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

et $u = U_M$. La matrice C est $(0 \ 0 \ 0 \ 1)$.

Ce modèle linéaire est utilisé pour la conception du régulateur.