PLATEFORME POUR L'ETUDE ET LA PROGRAMMATION DES ASSERVISSEMENTS ECHANTILLONNES

Eric DUQUENOY,
IUP du Littoral – Université du Littoral Côte d'Opale (ULCO)
40 rue F.Buisson - BP649
62228 Calais - FRANCE.
mailto:eric.duquenoy@iup-calais.univ-littoral.fr

Résumé : Nous présentons une plateforme logicielle ouverte permettant l'étude et la programmation d'asservissements numériques en langage C pour la réalisation de travaux pratiques en licence et en IUP.

Mots clés : asservissements échantillonnés, régulateurs numériques, langage C.

1. INTRODUCTION

1.1. Cadre de travail

Dans le cadre d'un enseignement en automatique linéaire, dispensé en année de licence dans la filière GSI[1] de l'IUP[2] de Calais ainsi qu'en licence EEA[3] à l'ULCO, nous avons mis au point une application ouverte, basée sur le principe de programmes sources modifiables, permettant aux étudiants d'implanter rapidement des algorithmes de régulation écrits en langage C tout en s'affranchissant des contraintes de développement graphique sous Windows, et proposant une interface conviviale de saisie des données de régulation.
Outre cet enseignement d'automatique linéaire, composé de cours, de travaux dirigés et de travaux pratiques couvrant les aspects continus et échantillonnés de ce domaine, les étudiants reçoivent également des enseignements de programmation en langage C et de traitement de signal, ce qui les rend à même de mettre en place des algorithmes de régulation.
Après avoir exposé les objectifs pédagogiques des travaux pratiques d'automatique, nous décrirons quelques solutions logicielles existantes puis nous présenterons les objectifs que nous nous sommes fixés ainsi que le matériel sur lequel nous avons travaillé, puis nous détaillerons notre application. Enfin, nous exposerons quelques résultats et nous conclurons.

1.2. Objectifs pédagogiques

Le but des séances de travaux pratiques d'automatique, est de mettre en oeuvre, à travers quelques processus différents (vitesse, position, température, niveau), les principaux régulateurs propres aux asservissements linéaires continus ou échantillonnés, comme les correcteurs industriels classiques (proportionnel – intégral – dérivé) ou les correcteurs adaptés (réponse pile, placement de pôles) dont les aspects théoriques sont vus en cours et en travaux dirigés. Après avoir étudié et pratiqué ces asservissements dans le domaine analogique, nous abordons la structure des régulateurs numériques notamment en proposant aux étudiants leur implantation dans un langage évolué. C'est sur ce point que porte plus précisément ce projet.

1.3. Solutions logicielles disponibles

1.3.1. Solutions généralistes

Hormis les correcteurs PID étudiés dans le domaine continu et donc réalisés avec des composants électroniques, tous les autres correcteurs nécessitent une approche numérique. Il est donc important de bien choisir la solution logicielle qui permettra d'aborder de manière exhaustive l'ensemble des correcteurs. Si des solutions généralistes telles que `Simulink` de `MathWorks` [1] ou `Labview` de `National Instrument` [2] permettent de réaliser, sous Windows, des asservissements de grande complexité, en revanche, ils nécessitent, de la part de l'étudiant moyen, une bonne maîtrise de leur utilisation avant de pouvoir être efficace. De plus, ces solutions nécessitent des cartes d'entrées-sorties propriétaires et, bien souvent, coûteuses.

1.3.2. Solution spécialisée

Une solution spécialisée dans les asservissements, comme `Acsyde` de `Ipsis` [3] permet, quant à elle, une prise en main rapide. Elle présente également l'avantage d'être compatible avec un nombre important, et relativement éclectique, de cartes d'acquisition, dont celle que nous possédons actuellement et décrite en section 2.

1.3.3. Position du problème

Cependant, toutes ces solutions ont en commun un écueil important : elles ne permettent que d'étudier l'aspect fonctionnel de l'asservissement, l'étude s'effectuant sous forme de schémas blocs, chaque bloc étant décrit, sous forme d'équation, par sa transmittance en *z* ou en *p*. La programmation de bas

[1] Génie des Systèmes Industriels
[2] `http://iupcalais.univ-littoral.fr`
[3] Electronique Electrotechnique Automatique

niveau d'un régulateur numérique, et donc la mise en évidence des problèmes qui lui sont liés (choix des types de variables, débordements numériques, durée de la boucle de régulation, …), est donc quasiment impossible avec ces logiciels.

1.4. Solution envisagée et cahier des charges.

Puisqu'il n'existe pas, à notre connaissance, sur le marché des logiciels pédagogiques ou industriels, d'application de ce type permettant d'aborder simplement les problèmes de programmation d'algorithmes de régulation sur PC en un minimum de temps, avec un langage très répandu comme le C et de plus adapté au matériel existant, nous avons donc développé notre propre plateforme. Les contraintes de développement que nous nous sommes imposées étaient les suivantes :

- la programmation doit s'effectuer à l'aide d'un langage connu de l'étudiant et courant dans l'industrie. Bien sûr, ce langage doit être de type compilé pour des problèmes de rapidité,
- l'étudiant doit pouvoir maîtriser toutes les facettes de la régulation :
  - choix de la fréquence d'échantillonnage,
  - élaboration de la grandeur d'erreur,
  - génération de consignes en échelon ou en rampe,
  - utilisation d'une consigne externe,
  - problèmes liés à la conversion analogique-numérique comme la résolution des convertisseurs ou le débordement numérique,
  - implantation de régulateurs quelconques tels que PID, réponse pile, tout-ou-rien, etc…
- récupération des mesures pour une exploitation graphique ou des post-traitements en vue d'identification par exemple.
- adéquation du logiciel avec le matériel existant avec des possibilités d'évolution vers d'autres appareillages. En particulier, un point important est la rapidité de la boucle de régulation même sur des machines relativement anciennes (PII-266Mhz). Cela exclut donc l'utilisation de langages interprétés comme celui de MATLAB par exemple.
- Faible encombrement mémoire de l'application de manière à pouvoir stocker des séquences de mesures relativement longues.

Le nombre de séances étant peu élevé et leur durée courte, il fallait convenir d'un outil de programmation où l'étudiant puisse se consacrer essentiellement à l'algorithme de régulation et n'en soit pas détourné par des problèmes purement informatiques de programmation graphique et d'interfaçage avec le matériel.

2. MATERIEL

Nous disposons, pour les travaux pratiques d'automatique, de bancs de régulation de marque LEYBOLD équipés pour l'étude d'asservissements de position, de vitesse, de température et de niveau de liquide. Chaque banc possède une interface composée d'une carte sur bus ISA permettant la liaison avec un ordinateur et d'un module d'acquisition permettant la connexion au reste du banc (figure 1).

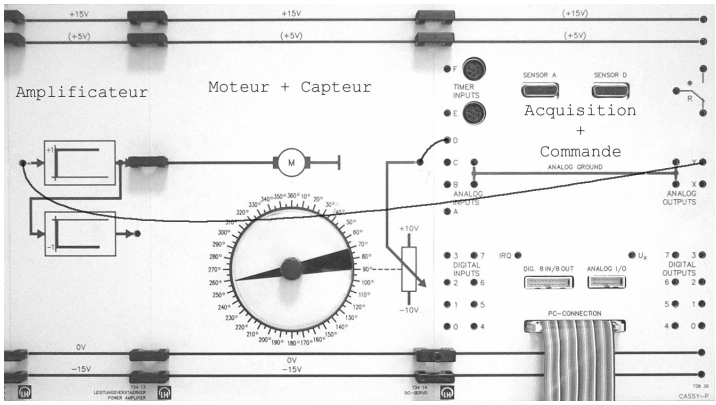


Figure 1 : Le module d'acquisition permet la connexion au reste du banc. Nous sommes ici dans le cas d'un asservissement de position où la sortie analogique X du module d'acquisition est connectée à l'entrée de l'amplificateur de commande du moteur. La sortie capteur du moteur est connectée à l'entrée analogique D. Le générateur de consigne (non représenté ici) est connecté sur l'entrée A.

L'interface permet la lecture simultanée de 4 entrées analogiques appliquées à des convertisseurs numériques-analogiques de 12 bits. Deux convertisseurs numériques-analogiques, codés également sur 12 bits, sont disponibles. L'interface offre également l'accès à 8 entrées et 8 sorties logiques compatibles TTL.

Tous ces éléments sont pilotables par le logiciel ACSYDE. Cependant, celui-ci ne permet pas de programmer les algorithmes de régulation. Il travaille sous forme de schémas blocs décrivant ainsi une régulation sous forme de transmittances. Il ne permet donc pas de mettre en évidence l'aspect algorithmique d'une régulation. Il fonctionne néanmoins sous Windows NT4 grâce à une DLL permettant l'accès aux fonctions d'entrées-sorties. En effet, sous ce système d'exploitation, il est impossible d'accéder directement aux ressources matérielles. Il faut donc passer par une autorisation du noyau, ce que permet la DLL en question. Cette DLL est fournie gratuitement par la société `Scientific Software Tools` [4].

Le matériel piloté par la version actuelle de notre application est constitué d'un banc de régulation de marque LEYBOLD équipé de :

- un générateur de consigne,
- une partie opérative (moteur, pompe, etc.),
- capteurs (position, niveau, etc.),
- l'amplificateur,
- une interface CASSY.

3. SOLUTION PROPOSEE

3.1.1. Outil de programmation

Nous utilisons ici le Visual C [6] de chez Microsoft compatible avec la DLL précédemment citée et qui autorise l'accès aux entrées/sorties sous Windows NT4/2000/XP. Notre plateforme reste cependant parfaitement compatible avec les versions Windows 9x puisqu'il ne s'agit que d'une redirection des commandes `inport` et `outport` du langage C.

3.1.2. L'application

Nous avons souhaité réaliser une véritable application et non un ensemble de fonctions qu'il faudrait assembler pour obtenir un programme fonctionnant correctement. Aussi, seules les sources des fonctions de régulation sont modifiables, le reste de l'application étant fourni sous forme de modules objets. Ainsi, même sans aucune modification de la part de l'étudiant des fonctions de régulation mises à leur disposition en début de séance, la compilation ne génère aucune erreur critique. De plus, seules quatre fonctions nouvelles ont été créées pour gérer l'interface : sortie vers un convertisseur, lecture d'un convertisseur, initialisation de l'horloge d'échantillonnage, synchronisation sur cette horloge. Ce principe facilite grandement le travail de déboguage des travaux des étudiants par l'enseignant responsable de la séance puisque la recherche des erreurs se limite aux parties modifiables.

3.1.3. Structure de l'application

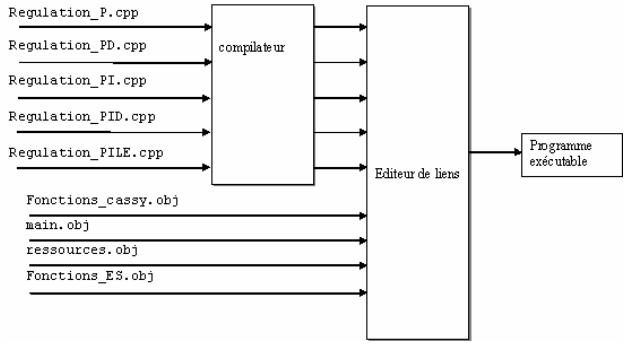


*Figure 2 : structure logicielle de la plateforme. Seuls les fichiers *.cpp sont modifiables. Ils contiennent les implantations des différents régulateurs.*

Pendant la séance de travaux pratiques, les étudiants ont à leur disposition l'ensemble de la plateforme constituée des fichiers suivants (voir figure 2) :

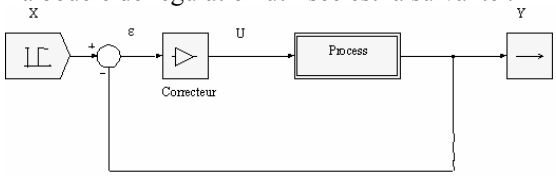
- Fichiers sources modifiables (`regulation_*.cpp`) dans lesquels l'étudiant implante ses algorithmes de régulation sous forme de fonctions appelées par le programme principal,
- fichier objet (`fonctions_cassy.obj`) contenant les fonctions propres à la carte d'acquisition, ici un module CASSY associé à sa carte interne,
- fichier objet (`ressources.obj`) contenant les ressources graphiques de l'interface,
- fichier objet (`fonctions_ES.obj`) contenant les fonctions d'entrées et de sorties (uniquement pour la programmation sous Windows NT, 2000 et XP)

Après compilation et lancement du programme, l'interface graphique qui apparaît, permet d'entrer les différents paramètres de la régulation tels que durée de la mesure, fréquence d'échantillonnage, type et paramètres du régulateur et emplacement de destination du fichier de mesures. Ces données sont ensuite transmises à la fonction de régulation.
Enfin, la validation de la régulation peut s'effectuer soit classiquement avec un oscilloscope pendant la mesure soit avec un logiciel externe en exploitant le fichier de mesure que génère notre application et dans lequel apparaît l'ensemble des signaux (entrée, sortie, erreur et commande). Dans cette perspective, nous avons développé également un utilitaire de visualisation de ce type de fichiers.

4. RESULTATS

4.1. Processus

Nous présentons ici, pour un asservissement de position muni d'un correcteur proportionnel, une comparaison entre les résultats de mesures obtenus d'une part avec une description fonctionnelle de l'asservissement sous `ACSYDE` et d'autre part avec notre plateforme.
La boucle de régulation utilisée est la suivante :



Le processus est un premier ordre avec intégrateur, pour une constante de temps de l'ordre de 55ms et un gain statique de 0.7 :

$$\frac{0.7}{p(1+0.055p)}$$

En boucle fermée, la transmittance est :

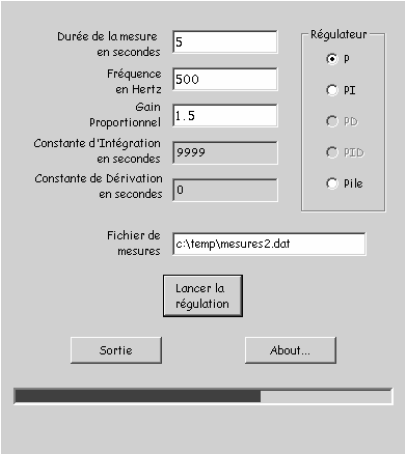
$$\frac{K_p * 0.7}{K_p * 0.7 + p + 0.055 * p^2}$$

Les réglages proposés sont les suivants :

gain proportionnel K_p :	**1.5**
durée de la mesure :	**5s**
fréquence d'échantillonnage :	**500Hz**

4.2. Mesures

Après lancement de notre application, l'interface graphique apparaît et nous sélectionnons le régulateur proportionnel, préalablement programmé dans le fichier `régulation_P.cpp` :



En fonction du régulateur choisi (boutons de droite), seuls les champs nécessaires sont accessibles. Ces paramètres sont ensuite transmis au régulateur programmé par l'étudiant.

4.3. Exploitation des résultats

Lorsque la régulation est terminée, c'est-à-dire après la durée spécifiée dans la boite de dialogue (ici 5 secondes), un fichier de mesure est écrit sur le disque. Celui-ci pourra être exploité par un programme externe. Cependant, aucun accès disque n'est réalisé pendant la régulation de manière à ne pas ralentir le programme. En effet les mesures sont stockées en mémoire vive : c'est la raison pour laquelle nous avons préféré utiliser un programme externe pour la visualisation des résultats de manière à donner à notre application le maximum de mémoire libre. De plus, dans la cadre d'un travail en binôme, comme c'est souvent le cas en travaux pratiques, cela permet aux étudiants de se partager le travail (mesures et exploitation) en utilisant des machines différentes.

4.4. Performances

L'expérience montre, qu'avec notre outil, les étudiants sont capables de réaliser la programmation d'un asservissement de position, écrit en langage C, avec l'ensemble des régulateurs proposés (P, PI, PID et réponse pile) en deux séances de 3 heures. Le résultat obtenu dans le cadre de notre exemple, pour une consigne externe de 5v, est donné figure 3. Le tracé a été réalisé avec notre utilitaire de visualisation. Nous donnons, pour comparaison, les résultats obtenus avec ACSYDE (figure 4 et figure 5).
En fonction de la complexité du régulateur, la fréquence d'échantillonnage peut être réglée jusqu'à 3kHz sur un PII-266Mhz.

5. CONCLUSION

Notre plateforme nous a permis d'aborder les différents aspects de la programmation de bas niveau des asservissements numériques. Nous envisageons plusieurs évolutions : adaptation à d'autres matériels, extension au filtrage numérique, commande à distance par réseau ethernet.
Enfin, nous remercions l'IUP du littoral pour nous avoir permis d'élaborer ce projet.

6. BIBLIOGRAPHIE

[1] Mathworks : http://www.mathworks.fr
[2] National Intruments : http://www.ni.com
[3] Ipsis : http://www.ipsis.com
[4] S.S.T. : http://www.driverlinx.com
[5] Cours d'automatique (tomes 2 et 3), *M.Rivoire et J-L.Ferrier*, Eyrolles, 1990
[6] Atelier Visual C++, *D.Kruglinski*, Microsoft Press, 1997

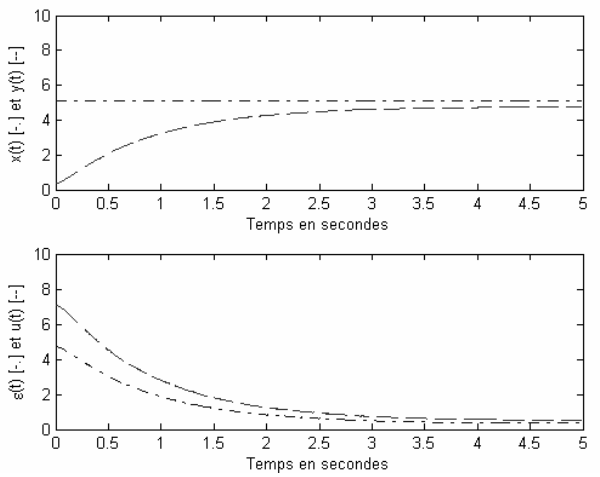


Figure 3 : Le processus commandé par notre plateforme. En haut la consigne x(t) et la sortie y(t). En bas l'erreur ε(t) et la commande u(t).Pour t=0, le rapport u/ε donne le gain K_p.

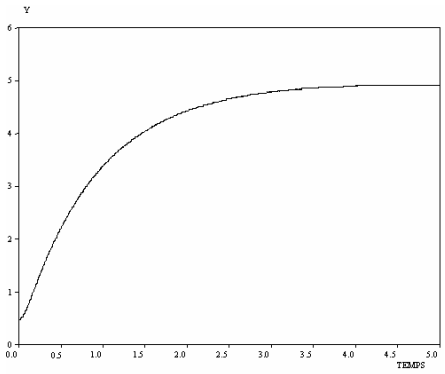


Figure 4 : signal de sortie y(t) du système mesuré avec ACSYDE. Le processus est du second ordre et la consigne est de 5v.

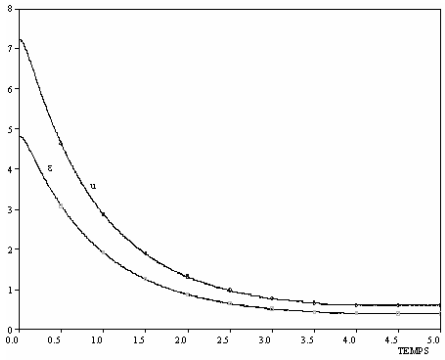


Figure 5 : signaux d'erreur et de commande sous ACSYDE