Ecole doctorale Sciences et Santé **Centre Européen pour la Recherche Nucléaire**

Algorithmes de Contrôles Avancés pour les Installations à Gaz du LHC au CERN suivant le Framework et l'approche dirigée par les modèles du projet GCS

THESE

Présentée et soutenue publiquement pour l'obtention du grade de

Docteur de l'Université de Picardie Jules Vernes

par

Sébastien Cabaret

le xx Mars 2008 devant le jury composé de :

Mr. Xxxxx xxxxxxx	Président
Mr. André BIGAND	Rapporteur
Mr. Xxxxx xxxxxxx	Rapporteur
Mr. Renaud BARILLERE	Examinateur
Mr. Xxxxx xxxxxxx	Examinateur
Mr. Ahmed RACHID	Directeur de thèse
Mr. Hervé COPPIER	Co-Encadrant de thèse

Laboratoire des Technologie Innovantes (LTI)

Un peu de science éloigne de Dieu, mais beaucoup y ramène.
Louis Pasteur

Remerciements

Table

Liste des Figures

Préambule

L'automatique est une science de l'ingénieur qui a pour but majeur de résoudre des problématiques d'analyse, de modélisation et d'identification en vue d'élaborer des commandes adaptées. Les outils industriels à dispositions s'appuient sur les technologies de programmation pour contrôleurs ou microprocesseurs qui assurent un traitement centralisé des automatismes. L'informatique industrielle est de l'informatique appliquée pour la conception des systèmes automatisés industriels. Elle assure la conception, la réalisation (programmes) et l'interfaçage de l'ensemble des éléments constitutifs des automatismes

Les automates programmables industriels quant à eux sont le cœur du système automatisé dans nombres d'automatismes d'aujourd'hui. Ils sont robustes, modulaires et répondent à des exigences techniques communément appréciées. Leurs structures et leurs softwares de développement sont soumis à des standards de diverses natures. Ils permettent le développement avec des langages normalisés et assurent donc une certaine homogénéisation des codes sources du contrôle-commande. Grace à cette convergence des langages de programmation, les applications automatisées en sont améliorés.

On parle de régulations avancées pour automates, les algorithmes de commande qui sont au-delà des méthodes classiquement mises à disposition dans l'automate. Les méthodes sont nombreuses et variées : elles ne répondent pas toutes aux mêmes spécificités et leurs synthèses peuvent être radicalement différentes. Leurs implémentations sont donc sujettes aux exigences des différentes approches de mise en œuvre.

La taille et la qualité requise des applications ont fait développer des solutions informatiques supplémentaires permettant la réalisation plus efficace des systèmes automatisés industriels. Par exemple l'émergence des frameworks ou 'canevas' de développement ont vu le jour pour répondre plus pleinement à ce souci d'optimisation. De ce fait, certains concepts venant du génie informatique inspirent l'informatique industrielle.

Travailler avec des outils et des solutions évoluées n'est pas nécessairement un gage de simplicité. Ceci n'est pas sans conséquences pour l'automaticien. Il doit toujours concevoir ses automatismes et faire de la régulation industrielle.

La problématique des travaux présentés ici est de deux ordres. Le premier enjeu réside en l'implémentation d'algorithmes avancés de régulation dans un automate programmable industriel utilisant les standards de programmations industrielles (norme CEI 61131-3).

Cet axe de travail s'accompagne d'études et de réalisations sous automates pour l'identification et la modélisation de processus à contrôler. Le deuxième enjeu de travail est attenant au précédent : il consiste en l'étude et l'utilisation de deux approches récentes de l'informatique industrielle (framework de développement et la réalisation dirigée par les modèles) en vue d'y inclure les régulations avancées pour automates.

Pour résumer, cette thèse a pour but de montrer la mise en œuvre pour un automate programmable industriel d'algorithmes de contrôles avancés, d'identification et de validation suivant ces nouvelles méthodologies de développement.

Enfin ces travaux de thèse s'inscrivent dans un contexte de recherche appliquée au CERN (Centre Européen pour la Recherche Nucléaire). Le projet concerné par la thèse est le projet LHC Gas Control System des Expériences du nouvel accélérateur.

INTRODUCTION

1. Objectifs de la thèse

2. Contexte de travail au CERN

3. Contexte de développement

4. Etat de l'art

5. Organisation de la thèse

6. Conclusion

Introduction

Objectifs de la thèse

Les travaux de thèse se sont effectués au CERN. Le Laboratoire Européen pour la Physique des Particules, est l'un des centres de recherche les plus prestigieux du monde. Son domaine d'activité est la physique fondamentale qui s'attache à découvrir comment fonctionne notre univers, d'où il vient et où il va. Le CERN utilise quelques-unes des machines les plus grandes et les plus complexes du monde pour étudier les plus minuscules constituants de la nature.

Cette thèse s'inscrit dans le cadre de l'analyse et de la conception de commandes avancées pour les systèmes à gaz des détecteurs et sous détecteurs des expériences du nouvel accélérateur construit au CERN (LHC : Large Hadron Collider).

Les études ont été élaborées au CERN en collaboration avec les équipes en charge de la fabrication, de la réalisation du contrôle-commande, de la mise en service et de la maintenance de l'ensemble des 23 installations à gaz concernées.

La particularité des travaux de thèse viennent du contexte fortement industriel des solutions utilisées et d'un environnement de recherche appliquée.

Les objectifs visés par la thèse sont résumés sous quatre aspects :

- Le premier objectif consiste en la synthèse et la mise en œuvre d'algorithmes de régulations avancées dans un outil industriel avec des contraintes fortes de développement et d'intégration. Les travaux suivent des approches sophistiquées du génie logiciel appliquées à l'informatique industrielle. Une partie supplémentaire est dédiée à l'identification et à la modélisation des systèmes à gaz.
- Le deuxième aspect réside par l'étude et l'acquisition des nouvelles orientations de développements pour la réalisation des applications contrôle-commande des processus gazeux des expériences du CERN. Cette étape est indispensable pour le succès de la thèse.
- Ensuite, le troisième sujet sous jacent à l'aspect précédent est l'étude des méthodologies Framework et Model Driven appliquées à l'informatique industrielle.

- Enfin l'objectif global de cette thèse est de répondre à un besoin fondamental des systèmes à gaz : mettre à disposition des experts process, des solutions de régulations permettant de résoudre des problèmes complexes non solvables avec le simple PID. Cet objectif sous entend la mise en œuvre d'un dispositif convivial d'utilisation des algorithmes et d'une sensibilisation des utilisateurs qui seront amenés à régler les processus gazeux des Expériences du CERN. Cette réalisation s'intègre aux exigences du Framework et du Model Driven.

Contexte de travail au CERN

Le CERN, l'Organisation Européenne pour la Recherche Nucléaire, fut créé en 1954 par une convention entre 12 états européens. L'objectif de cette convention était de mettre en commun les moyens humains, techniques et financiers de chaque pays afin de concevoir et d'utiliser de grands instruments pour la physique nucléaire et la physique des particules. Le CERN compte aujourd'hui 20 états membres et 8 états observateurs. L'organisation est située sur la frontière franco-suisse aux environs de Genève.

Le CERN emploie aujourd'hui environ 2500 personnes qui couvrent un large éventail de compétences et de métiers : physiciens, ingénieurs, programmeurs, techniciens, ouvriers qualifiés, administrateurs, secrétaires, etc... D'autre part, environ 6500 scientifiques, soit la moitié des physiciens des particules au monde, viennent au CERN pour leur travail de recherche. Ils représentent 500 universités ou instituts et plus de 80 nationalités.

En provoquant des collisions entre les infimes particules de matière, les physiciens démêlent les lois qui régissent la nature. Les machines du CERN sont des accélérateurs et des détecteurs de particules.

Le CERN est également un centre multi-compétences qui a permit de contribuer activement aux avancées techniques dans nombres de domaines. On trouve entre autre l'imagerie médicale et le World Wide Web. Les chambres proportionnelles multi-fils inventées en 1968 (par Georges Charpak, prix Nobel de physique 1992) pour détecter des particules sont désormais utilisées en médecine et en biologie.

Le World Wide Web a été originellement conçu et développé (à la fin de 1990 par Tim Berners-Lee) pour pouvoir partager à tout instant des informations entre physiciens travaillant dans différentes universités et instituts aux quatre coins du monde. Le Web a maintenant des millions d'utilisateurs aussi bien dans le milieu académique que commercial.

L'ensemble des accélérateurs du CERN est articulées autour de quatre machines principales. La plus ancienne, le synchrotron à protons (PS), construit dans les années 50, a été brièvement l'accélérateur le plus puissant du monde. Le supersynchroton à protons (SPS), construit dans les années 70, a été le théâtre du premier prix Nobel du CERN dans les années 80. Le grand collisionneur électron-positron (LEP) en service de 1989 à 2000. Mais ce dernier est resté limité en énergie (208 GeV dans le centre de masse atteint en octobre 2000) et en luminosité, alors que la nouvelle physique pourrait se situer, d'après la théorie, à une échelle d'énergie de l'ordre du TeV. La construction du LHC (Large Hadron Collider) fut donc entrepris.

Le collisionneur LHC est installé dans le tunnel du LEP dont la circonférence est de 26,7 km et qui est enterré à une profondeur variant entre 50 et 170 m.

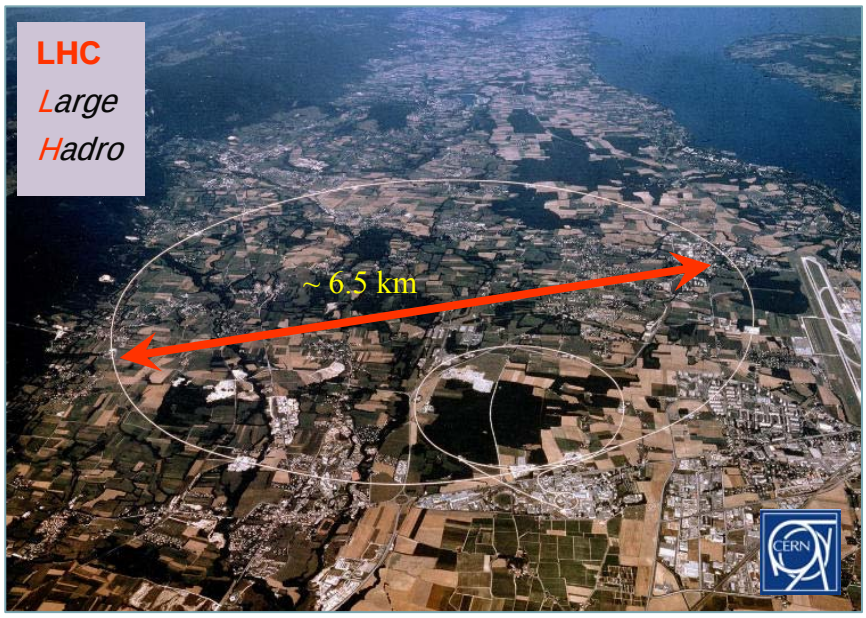


Figure 1 – Le LHC (Large Hadron Collider) dans le paysage Franco-suisse

Le LHC bénéficie de la chaîne d'accélérateurs du CERN : les protons sont accélérés par paquets dans l'accélérateur linéaire (LINAC2) jusqu'à 50 MeV, puis dans le Booster jusqu'à 1 GeV, ensuite dans le synchrotron à protons (PS) jusqu'à 26 GeV, et enfin dans le super synchrotron à protons (SPS) jusqu'à 450 GeV avant d'être injectés dans le LHC en deux faisceaux de sens contraire accélérés à une énergie de 7 TeV par proton, ce qui donne une énergie maximale dans le centre de masse de 14 TeV.

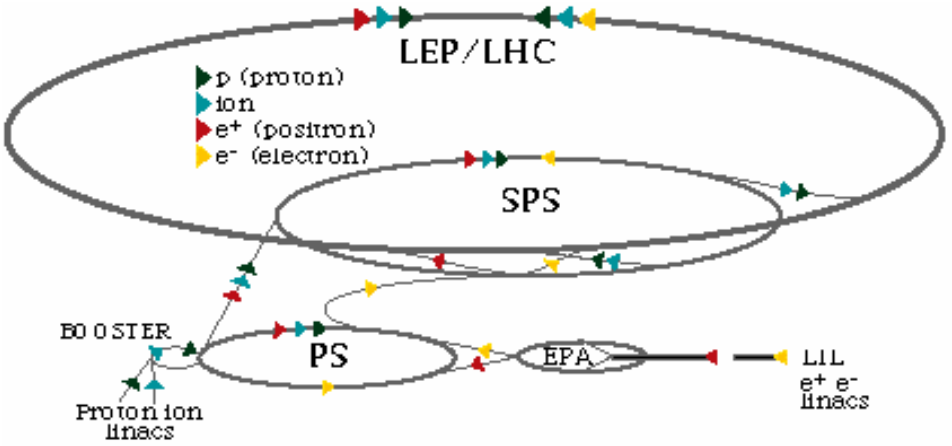


Figure 2 – Le principe d'accélérations successives des protons pour le LHC

Les performances d'un tel accélérateur ont deux grandes caractéristiques : l'énergie dans le centre de masse et la luminosité. Pour atteindre des énergies et une luminosité jamais obtenues jusqu'à présent, le LHC fait appel à une technologie de pointe dans de nombreux domaines.

Les deux faisceaux de particules du LHC circulent chacun dans un sens de l'anneau souterrain du LEP/LHC. Cet anneau circulaire se situe en grande partie en France et les particules peuvent collisionner en huit points, points où sont situés les sites d'Expériences : les quatre Expériences du LEP (Delphi, L3, Opal et Aleph) et les quatre Expériences du LHC (CMS, ATLAS, ALICE et LHCb).

Les Expériences du LHC ressemblent à des 'oignons' cylindriques, aussi gros que certains immeubles et remplis de systèmes électroniques sophistiqués. Les collisions se produisent exactement au centre de l'Expérience et les diverses couches du détecteur et sous-détecteur mesurent les différentes propriétés des particules émergentes.

Au plus près du faisceau se trouvent donc des détecteurs servant à enregistrer les traces des particules issues de la collision, et à suivre leurs trajectoires. Viennent ensuite les équipements de mesure de l'énergie, les calorimètres, dans lesquels la plupart des particules finissent leur parcours. La couche extérieure de l'oignon se compose elle aussi de trajectographes permettant d'identifier toute particule détectable arrivant aussi loin. Un aimant intègre dans le détecteur les trajectoires des particules chargées dans les trajectographes internes et contribue à l'identification des différents types de particules.

Chaque Expérience est composée de détecteurs et sous-détecteurs. Chacun d'entre eux est différent part son rôle et sa technologie. Par conséquent une Expérience n'est pas uniquement un assemblage de détecteurs mais plutôt un ensemble cohérent de

machineries, d'électroniques, d'informatique, etc., dédiés à la mise en œuvre de détecteurs et sous-détecteurs.

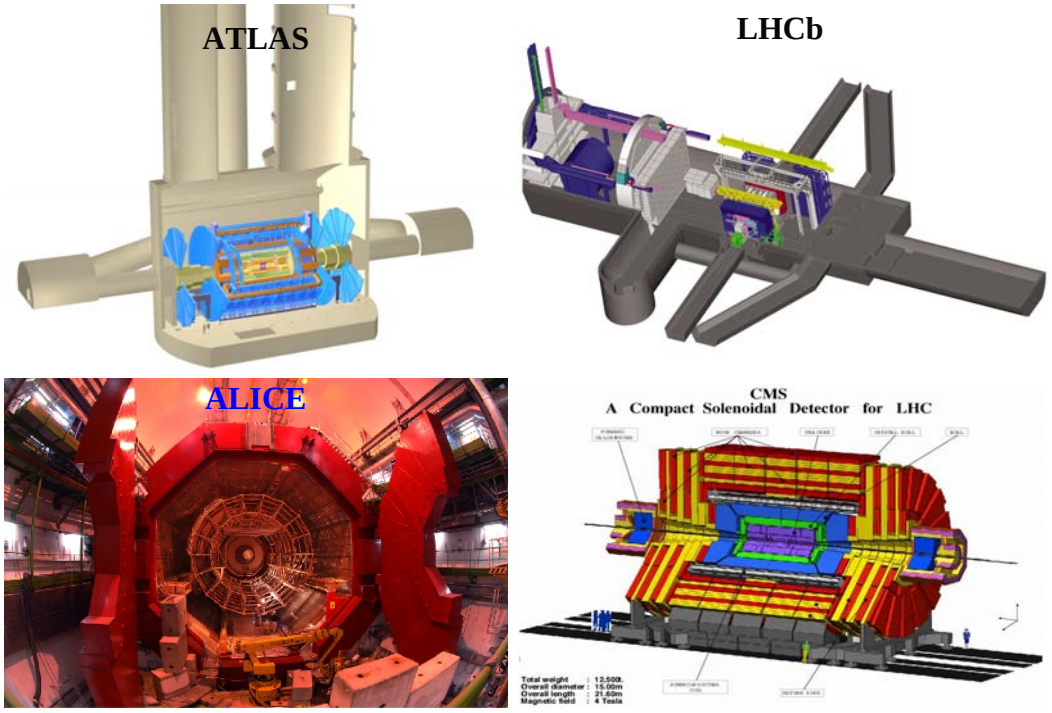


Figure 3 – Les quatre Expériences principales du LHC

Le contexte de travail de thèse au CERN concerne l'ensemble des systèmes de contrôle commande des sous détecteurs des Expériences nécessitant des mélanges gazeux. Cet ensemble représente 23 installations réparties sur les quatre Expériences du LHC.

Contexte de développement

Les automates programmables industriels (API ou PLC) s'attachent au contrôle des procédés en générant des signaux de contrôles et en acquérant des données directement sur le processus. L'architecture matérielle est classiquement composée d'une interface d'entrées-sorties, d'un processeur, de mémoires, et de modules de communication.

L'API est très souvent associé à une interface homme machine (Human Machine Interface (HMI)) et des organes externes. La figure qui suit résume simplement le fonctionnement de l'API dans une installation automatisée.

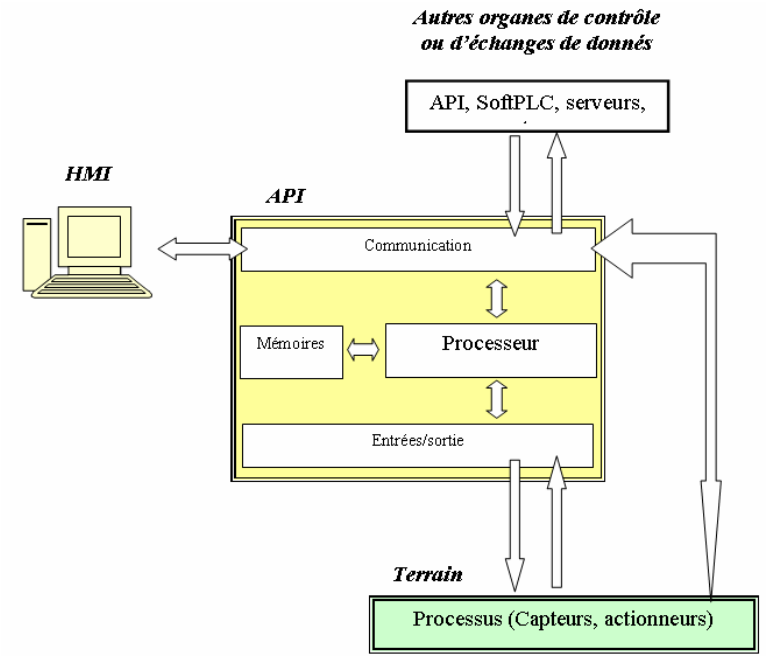


Figure 4 – L'API dans le système automatisé.

Remarque : on parle sans distinction d'automate programmable ou de PLC (Programmable Logic Controler).

Les travaux de thèse d'implémentation automate suivent la norme CEI-61131 relative à la spécification et à l'architecture des langages de programmation [Vei]. Cette norme se décline en sous-partie dont la norme CEI 61131-3 qui concerne les langages eux-mêmes et leurs éléments communs [Jim].

Les quatre langages standards automates décrits par cette norme sont :

- Le langage textuel de liste d'instruction IL (Instruction List)
- Le langage textuel de texte structuré ST (Structured Text)
- Le langage graphique de schéma à relais LD (Ladder Diagrams)
- Le langage graphique de diagrammes de blocs fonctionnels FBD (Functional Block Diagrams)

Ces langages sont reliés par les unités de programmation, ce qui signifie que toute implémentation dans un langage peut être 'utilisée' dans l'un des trois autres langages. Cette caractéristique forte de la norme autorise donc des stratégies de développement diverses pour assurer des fonctionnalités semblables. L'intérêt et le choix d'un langage par rapport à un autre est à l'appréciation du développeur automate.

Pour garantir la compatibilité, les quatre langages ont donc des éléments communs qui sont [Jim]:

- Les types de données et les variables (élémentaires ou structurées)
- Les Unités d'Organisation de Programme (UOP) qui communiquent grâce à des variables d'interface d'entrées/sorties ou des variables globales.
- Les diagrammes fonctionnels en séquences SFC (Sequential Function Charts).

Les travaux d'implémentation automates présentés dans cette thèse suivent la norme CEI 61131-3. Les 'objets' (UOP) ont été développés avec le langage ST. Les tests quant à eux utilisent fréquemment le langage FBD qui permet la visualisation rapide des variables et le forçage simple de quelques valeurs. La plupart des résultats ont été obtenus avec un automate programmable industriel type Premium de Schneider via le software de développement UnityV2.1.

Le projet de contrôle commande des systèmes à gaz (GCS : Gas Control Systems) utilise des méthodes nouvelles de réalisation. Le Framework de développement est un cadre défini pour la construction des applications de contrôle commande des 23 installations à gaz. Utilisant une approche objet dans l'automate programmable et dans le SCADA, le projet GCS est constitué d'outils logiciels dédiés aux automatismes. Une approche de développement dirigée par les modèles (Model Driven) est également utilisée pour une réalisation à un niveau supérieur des applications finales.

Etat de l'art

Les systèmes à gaz des détecteurs tels que ceux des Expériences du CERN ne sont pas nouveaux. Cependant les systèmes de contrôle utilisant des solutions industrielles n'existaient pas ou partiellement. La littérature à ce sujet est quasi inexistante et il n'y a pas forcément de documents pouvant servir de base aux études ici présentées.

Les systèmes de contrôle-commande des installations à gaz des Expériences du nouvel accélérateur sont une première réalisation en ce qui concerne les outils utilisés. Les automates programmables industriels s'étant fortement développés durant les années 90, il s'avère que cette solution d'automatisme soit nouvelle en ce qui concerne les détecteurs à gaz de particules.

Par ailleurs les solutions de développements tels que le Framework de travail et la réalisation dirigée par les modèles sont des stratégies totalement novatrices pour le domaine du contrôle-commande industriel.

Organisation de la thèse

La mémoire de thèse se décompose en quatre chapitres.

Le **premier chapitre** décrit le cadre du travail de thèse. Dans ce chapitre je m'attache à exposer en détail le contexte pratique des travaux de thèse. Je présente notamment le CERN, le nouvel accélérateur de particule en construction et les Expériences de physique fondamentale de la machine. Je m'attache ensuite à expliquer le projet Gaz des Expériences avec le contrôle-commande associé. Les structures et le design des installations à gaz sont aussi abordés afin d'exprimer les problématiques d'implémentation des régulations avancées.

Le **deuxième chapitre** s'intéresse aux techniques de modélisation et d'identification des processus à réguler. Au travers d'une étude synthétique sur les modèles, j'expose dans cette partie les bases nécessaires à la prise de décision pour la modélisation des processus gazeux des boucles de régulation du projet gaz des Expériences du CERN. J'aborde l'identification des processus avec comme objectif de montrer les résultats et les outils développés pour les automates programmables industriels du projet Gaz.

Le **troisième chapitre** est consacré à la correction des systèmes. Ce chapitre est la suite logique de l'identification. Je présente les divers algorithmes avancés de régulation que j'ai implémentés dans l'automate programmable. Dans cette partie du mémoire j'expose les résultats des algorithmes que j'ai développés sur un automate de marque Schneider (Premium sous UnityV2).

Le **quatrième chapitre** traite de l'intégration des algorithmes de contrôles avancés dans le framework de développement du projet suivant une réalisation dirigée par les modèles. Je réalise un développement à ce sujet afin d'apporter des enseignements sur l'influence de l'ingénierie informatique sur l'informatique industrielle. Dans cette dernière partie, je présente enfin les choix que j'ai faits pour intégrer des régulations avancées pour le projet Gaz. J'ai ainsi travaillé par le biais d'un objet que j'ai spécialement pensé pour répondre aux problématiques du framework et du développement dirigé par les modèles : le MultiController.

CHAPITRE 1

LES INSTALLATIONS A GAZ

1. Le projet LHC Gas Control System

2. Les systèmes à gaz

3. Les régulations

4. Le système de contrôle

Chapitre 1. Les installations à gaz

Dans ce Chapitre j'expose le contexte de travail de la thèse au CERN. L'objectif est de montrer la complexité des installations à gaz, qui nécessite de mettre en œuvre des solutions de contrôle commande efficaces (les régulations avancées faisant parties de ces solutions).

Ainsi j'aborde le projet dans lequel j'ai participé. J'explique le fonctionnement des installations ainsi que le Design des structures. Je détaille ensuite l'ensemble des boucles de régulation des systèmes avec leurs exigences. Enfin je conclus cette partie sur le système de contrôle industriel utilisé pour le projet.

1. Le projet LHC Gas Control System

Le développement et la maintenance des systèmes de contrôle des quatre Expériences du LHC du CERN requièrent des ressources non négligeables. Fort d'une grande expérience avec le LEP, le CERN fut contraint de remanier la façon d'aborder la construction du LHC en regroupant les compétences, en cordonnant les tâches d'une manière plus centralisée en vue d'une certaine homogénéisation des installations.

Le projet JCOP (Joint Control Project) fut créé au CERN dans le but de trouver des solutions communes de contrôle pour l'ensemble des Expériences du nouvel accélérateur. Dans ce contexte le projet LHC Gas Control System a été mis en place pour fournir l'ensemble des applications de contrôle pour les 23 systèmes à gaz du LHC.

2. Les systèmes à gaz

Les systèmes à gaz du LHC ont pour objectif simple de fournir aux détecteurs et sous-détecteurs des mélanges gazeux maintenus dans des conditions physiques de fonctionnement bien spécifiques. Pour résumer le fonctionnement de base, un système à gaz du LHC est conçu en boucle quasi-fermée pour la circulation du fluide. A partir des lignes primaires il lui faut un mélangeur pour ajuster la mixture demandée (Mixer). Le mélange gazeux doit être distribué dans les chambres du volume (détecteur).

Un système de Buffer (volume tampon) doit ajuster le flux en fonction des fluctuations atmosphériques et des aléas du process. Enfin le circuit fermé est mis en place grâce à une pompe de circulation. La pompe n'est cependant pas toujours indispensable pour les petits systèmes : le flux est alors assuré par un différentiel de pression maintenu le long du circuit à gaz.

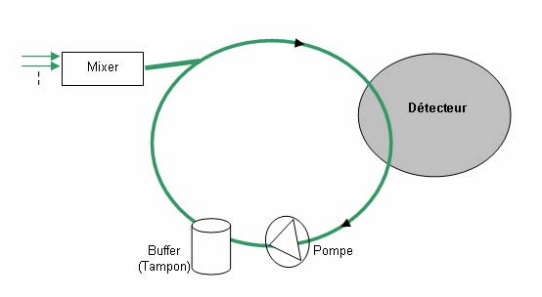


Figure 5 – Un système à gaz de base pour le LHC

Quelle que soit la conception du volume du détecteur, celui-ci n'est pas étanche à 100%. Ce constat simple explique les raisons de la circulation pour le maintien du mélange gazeux à des niveaux constants. Par exemple, de l'air pénètre dans le mélange interne gazeux du détecteur tandis que celui-ci « perd » de son mélange à l'extérieur. Il y a comme un équilibrage naturel qui se fait dans le temps.

Tout l'enjeu des systèmes à gaz du LHC est de préserver un mélange gazeux constant dans les conditions physico-chimiques voulues.

2.1. Structure et Design

Le flux du gaz est assuré par un différentiel de pression maintenu le long du circuit à gaz. Ceci est assuré par une conception mécanique adaptée (longueur, diamètres des lignes, pompe, vannes contrôlées…).
Les systèmes à gaz sont identifiés par des modules de fonctionnement permettant ainsi un traitement structurel efficace de chacun des 23 systèmes des Expérience du LHC. Par exemple le module Mixer assure le mélange et le débit des fluides gazeux. Le module Distribution s'occupe de la distribution du mélange au sous-détecteur. La Pompe permet d'assurer la circulation du gaz.

Certains systèmes possèdent des éléments supplémentaires qui se greffent sur la boucle du système. On peut trouver ainsi un module de Purification, un module d'Analyse des Gaz, etc.

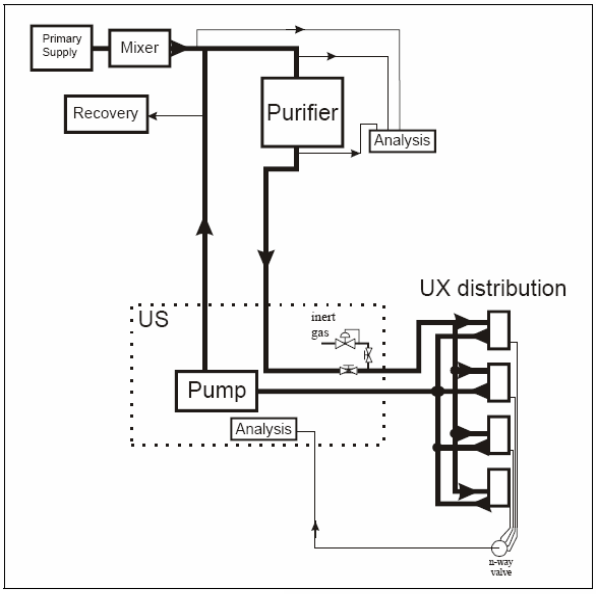


Figure 6 – Un système à gaz typique des Expériences du LHC

2.2. Fonctionnement

2.2.1. Le module des gaz primaires

Généralement il existe un module pour chaque Expérience du LHC. Les gaz sont stockés dans des bouteilles, des batteries de bouteilles ou des dewars. Ce stockage est localisé dans le bâtiment de surface et séparé du matériel informatique et électrique (Les dewars sont habituellement stockés en plein air). Par ailleurs des équipements sophistiqués sont installés à proximité (chromatographe, monochromateur, etc.). Chaque gaz primaire est équipé de deux sources d'alimentation différentes qui assurent un basculement automatique en cas de stockage vide pour l'un ou l'autre. Le contrôle des gaz primaires est monitoré par des pressions, des switchs de sécurité et des paramètres de contrôle permettant le suivi des variables. Un contrôle spécial est installé pour les gaz qui doivent être chauffés au niveau des bouteilles.

2.2.2. Le module pour le mélange (Mixer)

Le module Mixer est présent pour chaque système à gaz. Le processus est passif. Le mélange peut gérer jusqu'à trois lignes de gaz primaires. Il régule les débits des lignes pour arriver à des ratios désirés. Le Mixer mélange les gaz de manière turbulente. Tous les Mixers sont dans le bâtiment de surface et séparés du matériel informatique et électrique pour des raisons de sécurité.

Le mélange est transporté directement par un tuyau unique au détecteur qui se trouve dans la caverne (100 mètres plus bas). Le débit dépend de la demande du détecteur et des modules du système à gaz concerné. Le Mixer essaye également de maintenir une pression de fonctionnement constante.

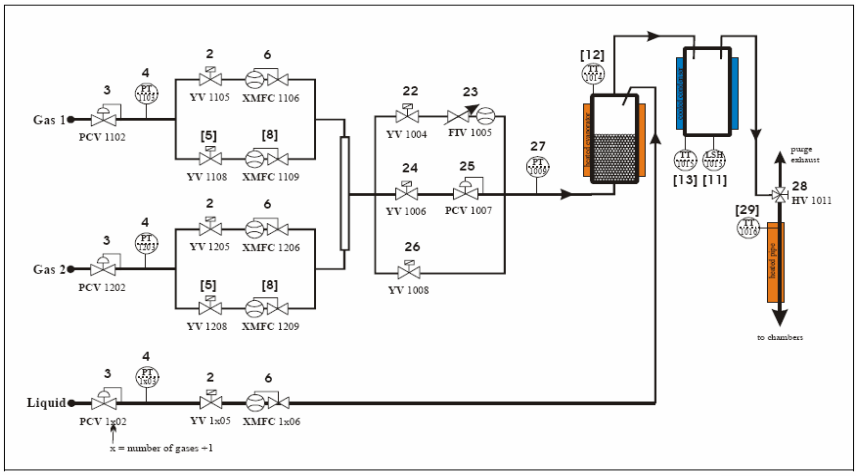


Figure 7 – Le module de mélange (Mixer) d'un système à gaz

Les principaux paramètres de contrôle du module sont des ratios, des débits, des détections de présence de gaz sur les lignes et des instruments indépendants de contrôle de composition du gaz. Ces instruments sont partagés pour plusieurs systèmes et peuvent servir de données de correction pour la mixture du détecteur.

2.2.3. Le module de Distribution

Le module Distribution est présent pour chaque système à gaz. L'arrivé du gaz correspond à la sortie du module Mixer. La Distribution est décomposée en racks. Chaque rack représente une enceinte du détecteur (jusqu'à 26 racks par détecteur). Chaque enceinte est décomposée en plusieurs chambres (jusqu'à 36 chambres par rack et jusqu'à plus de 250 chambres pour le plus gros système). La sortie de la Distribution se concentre en un seul tuyau. Les racks de distribution sont sous terre dans l'Expérience et sont totalement inaccessibles et toujours séparés de l'électronique et des ordinateurs.

La régulation de la pression interne du détecteur peut se faire sur la ligne de sortie de la Distribution. Les chambres sont assimilées à des canaux. Chaque sortie de canal peut être connectée à la chaîne d'analyse qui mesure la qualité du mélange.

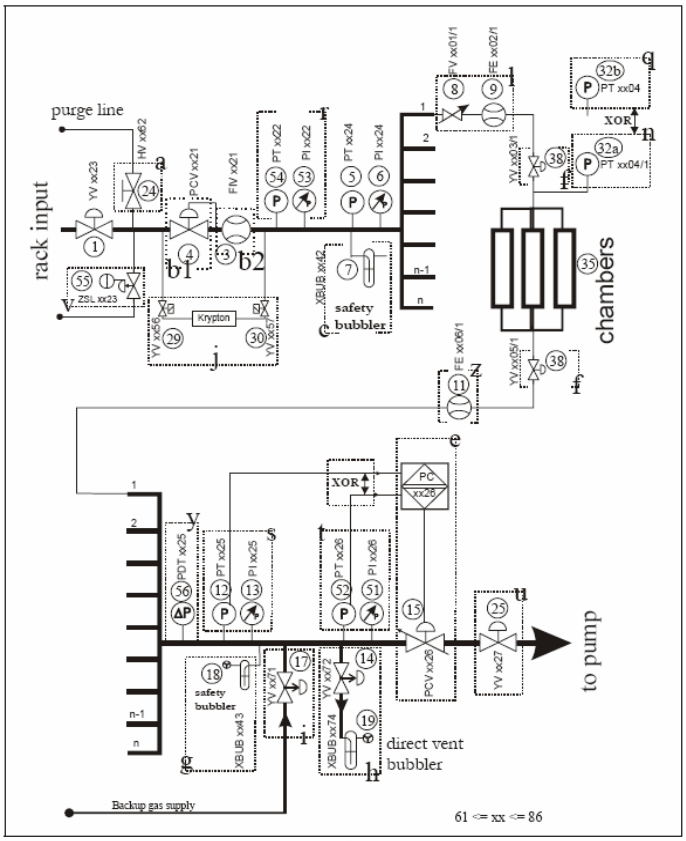


Figure 8 – Vue d'un rack du module de Distribution d'un système à gaz

2.2.4. Le module Pump et le module Exhaust

Le module Pump permet la circulation du gaz dans le circuit. C'est un élément actif situé en surface qui est principalement constitué d'une pompe. Ce module est. Son action se localise en sortie du détecteur et fait le lien entre la haute pression et la base pression. La régulation de la pression du détecteur peut s'effectuer à cet endroit suivant les configurations. Ce module peut également avoir plusieurs architectures (pompe tout ou rien ou analogique).

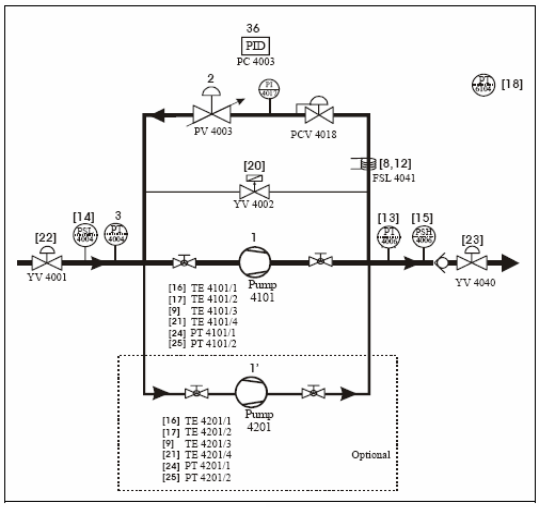


Figure 9 – Le module de recirculation (Pompe) d'un système à gaz

Le module Exhaust permet l'évacuation de la mixture gazeuse du circuit principal. Cette fonctionnalité est optionnelle en fonction des systèmes. Cela permet souvent une flexibilité d'opération supplémentaire. L'Exhaust est logiquement situé en surface.

2.2.5. Le module de purification et d'extraction CO_2

Le module de purification (Purifier) sert à certains systèmes qui ont la nécessité de palier à la détérioration de la qualité du gaz. Ce module est composé de deux colonnes pouvant fonctionner en alternance sur le circuit. Ce module est localisé en surface et sur la haute pression pour une meilleure efficacité de purification.

La purification est réalisée grâce à un agent chimique et une grille réactive permettant l'extraction de l'eau et de la contamination de l'oxygène. Lorsque la colonne concernée est saturée celle-ci devra passer par une phase de régénération à haute température. Pour avoir une opération et un fonctionnement optimal, les deux colonnes assurent une purification par leur alternance de fonctionnement (régénération). Le contrôle de ce module est essentiellement assuré par des capteurs de température et de qualité du gaz en entrée et en sortie. Ce module a pour particularité d'avoir des séquences de fonctionnement conséquentes qui permettent l'alternance du fonctionnement Process des colonnes.

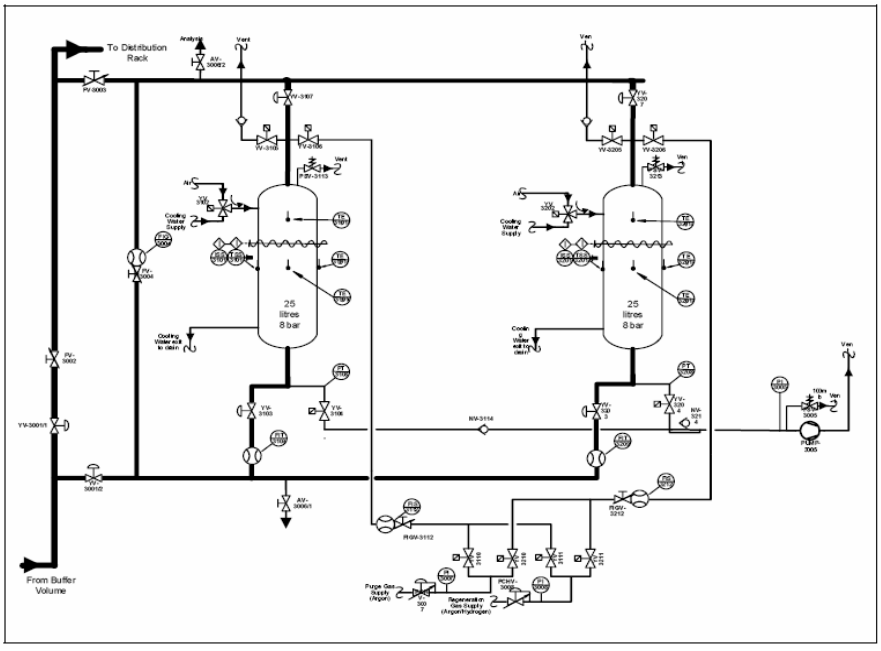


Figure 10 – Le module de Purification (Purifier) d'un système à gaz

Le module d'extraction du CO_2 (CO_2 removal) est réalisé sur le même schéma que le module de purification. Il permet l'extraction du CO_2 dans la boucle du circuit. Il est

également composé de deux colonnes pouvant fonctionner en alternance. Ce module est localisé en surface et sur la haute pression pour une meilleure efficacité d'extraction. La principale différence entre ce module et le module de Purification est la période de fonctionnement : le CO_2 removal est en état de marche sur quelques semaines (redémarrage annuel) alors que le Purifier marche quasiment en continue. Le contrôle d'extraction CO_2 est également similaire à la purification avec des séquences de fonctionnement complexes. Les éléments sont construits sur le même schéma avec des agents actifs différents et quelques équipements optionnels.

2.2.6. Le module d'analyse

Le module d'analyse consiste en des équipements d'analyse de mixtures de gaz provenant de divers endroits du système gaz. Le module est dans le bâtiment de surface. Il est organisé en chaînes d'analyse qui peuvent être décomposées en deux types. Le type A correspond à une sélection des lignes à analyser par des vannes pneumatiques standards. Le type B correspond à une sélection des lignes à analyser par des vannes spéciales à N-voies. Le contrôle de chaque chaîne est indépendant.

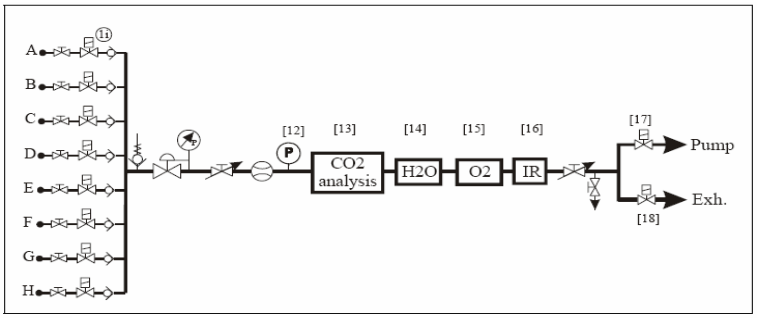


Figure 11 – Le module d'analyse de type A d'un système à gaz

2.2.7. Le rôle du Buffer

Le Buffer permet la compensation de la différence de pression atmosphérique. L'objectif du système à gaz est d'avoir le détecteur à pression ambiante. Toute infime variation de pression doit être compensée (régulation de pression). Le détecteur est dans une enceinte fixe et rigide qui ne permet pas l'expansion ni la compression du gaz. Si la variation est trop importante, le détecteur peut se détériorer. Le Buffer encaisse les surpressions du détecteur; si le détecteur est en sous pression celui ci 'fournit' (artificiellement) le volume de gaz nécessaire pour ramener le détecteur à l'équilibre.

3. Les régulations

Les 23 systèmes à gaz des Expériences de l'accélérateur sont structurés sur un patron commun (l'approche modulaire) et sont par ailleurs tous différents. Il est donc complexe de définir en détail les boucles de régulations qui ont une influence certaine sur chaque système à gaz.

Les systèmes à gaz définis précédemment ont pour mission de fournir une mixture gazeuse dans des conditions de fonctionnement prédéfinies. Le design et les éléments constitutifs des installations assurent par eux-mêmes des conditions favorables d'exploitations sans être pour autant suffisantes pour satisfaire l'ensemble des besoins. L'asservissement de certaines grandeurs physiques est indispensable. On peut distinguer plusieurs régulations dites 'critiques' pour le bon comportement du système à gaz.

3.1. Asservissement en cascade du débit des gaz primaires dans le Mixer

Cette boucle de régulation est essentielle car elle doit assurer un débit suffisant pour obtenir un ratio constant du mélange gazeux. Cette régulation en cascade est assurée selon le principe ci dessous :

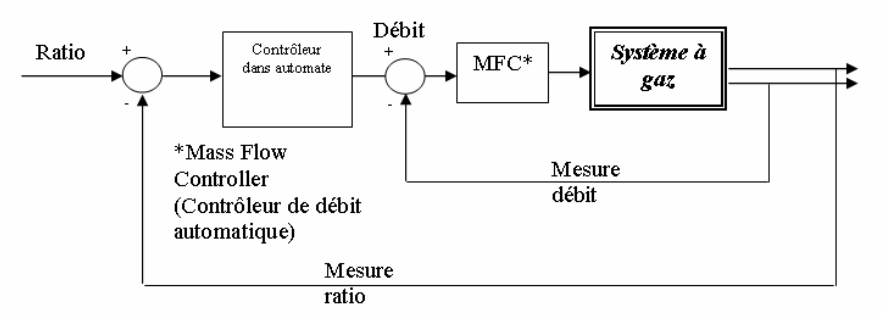


Figure 12 – Boucle de régulation du ratio dans le module Mixer

3.2. Régulation en pression de la sortie du détecteur - asservissement de la pression en générale (module Pump, Distribution et Exhaust).

La pression est sans aucun doute la valeur physique la plus difficile à maintenir aux vues des performances exigées. Pour que la détection des particules soit bonne, la pression des détecteurs doit être constante avec une précision souvent très proche du millibar. La moindre variation de pression a un effet sur la célérité des particules traversant le sous-détecteur. Ce phénomène est très ennuyeux pour la pertinence des mesures faites par les physiciens d'où la nécessité absolue d'assurer une pression correcte. Chaque système à

gaz a différentes stratégies (correspondant à leurs configurations) pour fournir une pression constante dans le détecteur.

La première façon est de contraindre la pression de sortie (PTxx25) en contrôlant l'ouverture des vannes de régulation en sortie des racks de la Distribution.

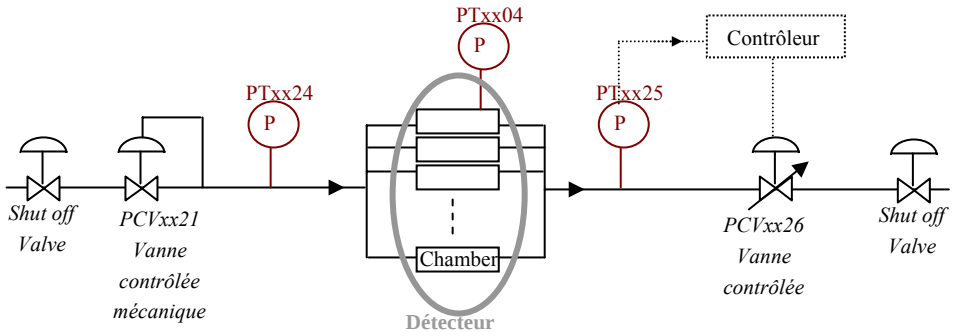


Figure 13 – Contrôle de la pression du sous-détecteur via la vanne de régulation en sortie des racks du module de Distribution

La deuxième manière de procéder est de contraindre la pression au niveau du module Pump. Cette stratégie utilise la vanne de régulation en by-pass avec la pompe. En jouant sur l'ouverture de la vanne, le débit de gaz en sortie du détecteur (et donc de l'entrée du détecteur) en est affecté. La pression en entrée (PT4004) peut ainsi être modulée via la vanne de by-pass.

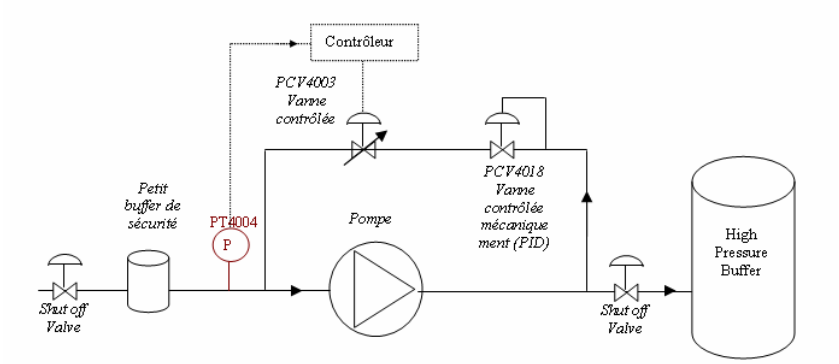


Figure 14 – Contrôle de la pression du sous-détecteur via la vanne de by-pass du module Pump

Il est également possible de combiner les deux vannes (sortie rack de Distribution et by-pass de la pompe) pour contrôler la pression.

Les deux stratégies de contrôle de pression du détecteur ne sont pas parfaitement isolées de toutes perturbations extérieures. Il est par exemple à noter que si le module d'Exhaust existe, celui-ci joue un rôle actif (régulation cascade sur MFC) sur la pression en sortie du buffer. Cette pression du buffer peut également être affectée par les variations de pression atmosphérique. De cause à effet une légère variation de pression du buffer implique des perturbations sur la pression du détecteur lui-même !

Le moindre changement de débit ou de configuration (nombre de rack connectés, etc…) influe sur la pression interne du détecteur. La régulation de pression est sans conteste la plus complexe et la plus difficile à mettre en œuvre.

3.3. Asservissement en température des colonnes du module Purifier et du module CO_2 absorber.

Le module de Purification (et/ou d'extractions de CO_2) n'est pas présent pour chaque système. Il possède cependant une régulation de température qui nécessite une certaine attention. En phase de régénération d'une colonne, la cartouche contenant l'agent actif doit être chauffée jusqu'à une température limite pendant une période de temps suffisante. Pour ce faire chaque cartouche est munie d'un fil chauffant et de trois capteurs de température afin de chauffer suffisamment l'intérieur contenant l'agent sans endommager l'extérieur.

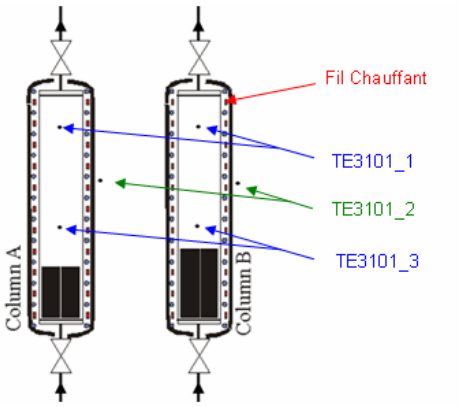


Figure 15 – Contrôle en température des cartouches du module de purification

3.4. Asservissement en humidité dans le module Humidifier.

L'humidificateur est un module qui a pour rôle d'introduire la quantité appropriée d'eau dans le processus à gaz. Le module régule le taux d'humidité via le débit d'eau (utilisation d'un MFC). Tous comme le module Mixer, l'asservissement en humidité s'effectue grâce à une régulation en cascade.

4. Le système de contrôle

Le système de contrôle du projet Gaz utilise des solutions industrielles. Ce choix arbitraire permet de concentrer la réalisation du contrôle-commande sur la mise en œuvre et le développement.

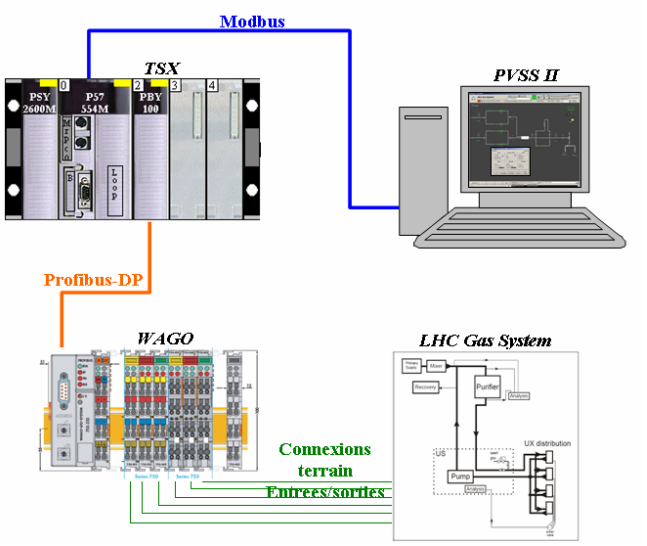


Figure 16 – Le système de contrôle d'un système à gaz des Expériences du LHC

Les systèmes de contrôles utilisent des automates programmables industriels de la marque Schneider (type TSX Premium). Le SCADA associé, fourni par ETM, est PVSS II. La communication entre ces deux derniers organes s'effectue via le protocole Modbus TCP. L'interfaçage entre le terrain et l'automate s'organise grâce à des modules déportés de marque WAGO. Les modules d'entrées-sorties (locales ou déportées) Schneider ne sont pas utilisés et la communication entre les modules WAGO et le TSX Premium est assurée par un module de communication Profibus-DP au niveau du châssis de l'automate.

5. Conclusion

Il y a 23 installations à gaz pour les Expériences du LHC. Le projet GCS a pour mission de réaliser le contrôle commande de ces systèmes avec des solutions industrielles. Les divers éléments constitutifs d'un système à gaz nécessitent de répondre à des contraintes physiques et techniques propres. C'est en ce sens que les travaux de thèse répondent à un besoin du projet : mettre en place des boucles de régulation efficaces permettant le bon fonctionnement des 23 installations.

L'enjeu des algorithmes avancés est de répondre aux exigences des experts à gaz (dynamique, robustesse, etc…). Pour obtenir de bons résultats, les travaux de thèse s'inscrivent dans les méthodologies de réalisation du projet. Concrètement, la problématique revient à donner aux ingénieurs de processus, des solutions de contrôles avancés pour satisfaire les exigences en régulations de leurs systèmes. Pour le projet GCS, c'est créer un 'service' de contrôle avancé avec une systémique d'implémentation dans le contrôle commande.

CHAPITRE 2

MODELISATION ET IDENTIFICATION DES SYSTEMES A GAZ

1. Introduction

2. Les modèles

3. L'identification du modèle pour le projet Gaz

4. Implémentation et outils d'analyse pour les systèmes à gaz

5. Application : résultats d'identification en ligne sous automates pour la modélisation et l'identification de la pression Détecteur d'Alice TPC

6. Conclusion

Chapitre 2. Modélisation et identification des systèmes à gaz

Dans cette partie, je traite de l'identification et de la modélisation des systèmes pour la détermination de stratégies spécifiques pour les installations à gaz des Expériences du LHC. J'aborde ainsi brièvement la notion de modélisation du point de vue général pour ensuite converger vers l'identification des modèles dans le contexte du projet GCS.

1. Introduction

La littérature sur la modélisation et l'identification système est aussi ancienne que pour le contrôle. Les premiers papiers majeurs dans les années 30-40 sur les réponses fréquentielles par Nyquist et Bode montrent cet intérêt précoce. Les travaux d'identification de Ziegler-Nyquist sur l'étude des réponses indicielles datent des années 40. Par ailleurs les progrès réalisés dans l'identification adaptive dans les années 60 ont considérablement contribué au développement des études dans le domaine. Les recherches s'organisent et en 1967 l'IFAC (*Symposium on Identification and System Parameter Estimation*) débute une série de symposiums qui produisent un nombre considérable d'articles sur les aspects et les problèmes de l'identification système. Aujourd'hui de nombreux livres et articles traitant de la modélisation et de l'identification sont disponibles et donnent des indications pratiques (ex: [Eyk], [Lan], [Bor]).

Avant de parler de modèles et d'identification, on parle volontiers de système. L. Ljung [Lju] explique que si l'on devait expliciter le terme 'système', on pourrait le définir comme étant un objet auquel différentes interactions produisent des réactions observables. Il ajoute que la détermination de modèles par l'observation et l'étude des propriétés propres à un système est l'essence même de la science. Il est effectivement remarquable de réaliser que la plupart des investigations scientifiques depuis leurs débuts n'ont eu pour but que de trouver des modèles de représentation suffisamment justes pour décrire les phénomènes naturels. La vision d'un modèle vrai est donc faussée au regard d'une démarche (arbitraire) qui définit un modèle pour un système quel qu'il soit. Ces dernières considérations d'ordre philosophique soulignent le principe relatif de modèle et de sa pertinence vis à vis d'un système réel. Toutes les démarches qui seront exposées prennent en compte ces hypothèses.

En se plaçant au niveau d'un référenciel contrôle-commande, la détermination d'un modèle est développée en vue de l'élaboration du système de commande [JMF]. En pratique, la construction d'un modèle s'effectue grâce à la connaissance et à l'observation des données du système soumis à des stimuli (entrées) et à ses réactions (sorties). L'expérience est également un facteur déterminant dans ce processus. Le modèle en automatique industriel a pour objectif de décrire le comportement d'un système en vue de la conception et de la mise en œuvre pratique d'une commande [Bor]. Pour ce faire, l'identification s'attache à la détermination des caractéristiques d'un modèle, ce qui revient à décrire de manière mathématique le comportement dynamique et stationnaire (si il y a lieu) d'un système. L'identification se résume donc à l'étude et à la conception mathématique d'un modèle suivant l'observation, la connaissance et l'expérience acquise sur le système.

L'identification système peut être considérée avec six traitements différents [Gra76]:

- ***La distinction entre système linéaire et non linéaire.***

Tous les systèmes sont naturellement strictement non linéaires. Un système est donc linéaire dans la mesure où pour une zone de fonctionnement, on le considère suivre un comportement quasi-linéaire. En linéaire, les principes de superposition sont applicables ce qui confère une aisance relative pour l'identification système.

- ***La différence entre système stationnaire et non stationnaire.***

La non stationnarité n'est pas à confondre avec la non linéarité. On parle de système stationnaire d'un système qui dans une plage de fonctionnement donnée avec les mêmes conditions physico-chimique ne voit pas ses paramètres varier. Cette propriété est fondamentale dans la modélisation et donc pour l'identification. Par simplification, on considère qu'un système est stationnaire dans la mesure où sur le temps nécessaire du relevé d'information pour effectuer l'identification, le système est stationnaire.

- ***Les systèmes continus ou discrets.***

Cet aspect du problème est rarement pris suffisamment en considération. Bien que le passage d'une formulation continue à une formulation discrète soit sans difficulté, les implications sur le traitement des données et sur la conception de la commande sont fortes. Le traitement sur la stabilité et la robustesse des systèmes corrigés est différent suivant que l'on évolue dans un environnement strictement continu ou non, (Les automates programmables industriels travaillent en discret).

- ***Le traitement mono ou multi variables.***

Le traitement théorique du problème mono ou multi variables peut être similaire sous certaines conditions. La problématique des systèmes multi variables (MIMO) réside dans

les techniques de réalisation concrète de l'identification (algorithmes et programmes complexes). Cependant les techniques d'identification mono variables restent significativement plus simples.

- ***Le caractère déterministe ou stochastique d'un processus.***

En pratique les mesures sont contaminées par du bruit ou part des problèmes de lissages/filtrage mal adaptés. On parle d'identification déterministe dans la mesure où l'on assume que les filtres, les bruits et autres perturbations sont compensés ou pris en compte. La réalité démontre fréquemment que ce postulat doit être pris avec réserve.

- ***Le degré de connaissance a priori du système.***

Ce dernier point est vraisemblablement le moins à négliger. La connaissance et l'expérience sont les meilleurs vecteurs de réussite. Ils permettent d'orienter sur les choix de modélisation et sur les stratégies d'identification.

Dans ce qui suit, nous ne traiterons que les systèmes déterministes et stationnaires.

2. Les modèles

Dans ce chapitre je me réfère aux divers modèles qui permettent de développer et de concevoir le système de commande. Pour ce faire, les modèles intéressants sont toujours de nature à représenter la dynamique et le régime établi. En opposition, les modèles de nature *statique* peuvent apporter des éléments intéressants comme la recherche de zone de fonctionnement.
Il existe un grand nombre de types de modèles *dynamiques*, chacun pouvant être destiné à des applications précises. On peut néanmoins discerner deux familles:

- ***Les modèles de connaissance basés sur les lois de la physique et de la chimie.***
- ***Les modèles de comportement basés sur les données d'entrées/sorties***

2.1. Les modèles de connaissance

Ces modèles sont élaborés à partir des lois de la physique et de la chimie. Les modèles de connaissance assurent une description assez complète des systèmes. Ils sont très utiles pour la simulation de processus.

La mise en équation d'un modèle de connaissance nécessite de définir les phénomènes physiques que l'on souhaite suivre [JMF]. Concrètement cela revient à choisir les variables pertinentes qui représentent le processus à contrôler. Toutes les variables

susceptibles d'intervenir et/ou de perturber le système doivent être définies et mises dans le modèle de connaissance.

Vient ensuite l'analyse précise de chaque variable avec sa fonction propre: variables d'état, variables d'entrées (manipulées) ou de sorties (mesurables ou non mesurables), variables de perturbations (mesurables ou non mesurables).

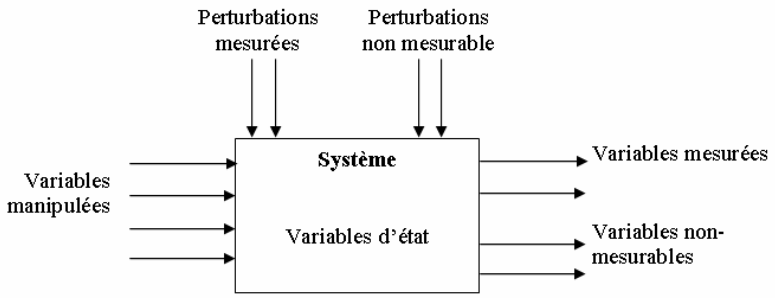


Figure 17 - Les variables du modèle de connaissance

Le principe du modèle de connaissance est de trouver les relations liant l'ensemble des variables en vue d'établir un ensemble d'équations mathématiques.

La ***Représentation d'Etat*** des systèmes est un outil puissant permettant de modéliser le fonctionnement de systèmes linéaires ou non, mono ou multivariables, en temps continu ou en temps discret et qui possède en outre, l'avantage de conserver la représentation temporelle des phénomènes [Gra03]. Cette représentation a comme avantage d'être bien développée pour la résolution et l'étude des problèmes d'automatiques a base de modèle de connaissances.

Exemple : on prend l'exemple de la figure ci-dessous avec (x_1, x_2, x_3) signaux 'internes' au système. On peut naturellement concevoir de commander ces différents signaux par l'intermédiaire du signal d'entrée e. Le problème de la commande du système peut donc se faire par la maîtrise simultanée de l'évolution de (x_1, x_2, x_3).

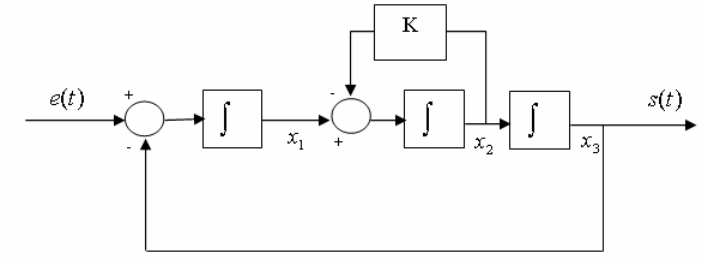


Figure 18 – Exemple : système et représentation d'état

On dit que l'ensemble de ces trois signaux forme l'état du système. Les variables internes sont appelées variables d'état du système. L'état d'un système est représenté par le ***vecteur d'état*** $\underline{x}(t) = \underline{x} = [x_1 \quad x_2 \quad x_3]$.

Pour un système à n variables d'état, m le nombre d'entrées et p le nombre de sortie, la représentation d'état est définie telle que :

$$\begin{cases} \dot{\underline{x}}(t) = [A(t)].\underline{x}(t) + [B(t)].\underline{e}(t) \\ \underline{s}(t) = [C(t)].\underline{x}(t) + [D(t)].\underline{e}(t) \end{cases}$$

[2.1. 01]

$$\begin{cases} A = n \times n \\ B = n \times m \\ C = p \times n \\ D = m \times p \end{cases}$$

[2.1. 02]

La première équation est ***l'équation de commande*** et la seconde, ***l'équation d'observation.*** Lorsqu'un système est modélisé sous la forme d'une représentation d'état, on montre qu'il est possible d'exprimer l'état du système à un instant donné en fonction du signal d'entrée à ce même instant et en fonction de son passé (état précédent). Pour l'exemple précédent, la représentation d'état devient:

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 \\ 1 & -K & 0 \\ 0 & 1. & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} e(t) \\ s(t) = [0 \quad 0 \quad 1] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \end{cases}$$

[2.1. 03]

2.2. Les modèles de comportement

Les modèles de connaissance sont très souvent complexes et malheureusement difficilement utilisables pour traiter les problèmes de modélisation des boucles de régulation des systèmes à gaz. Le modèle de comportement ou modèle expérimental est quant à lui beaucoup mieux adapté. La détermination de ce type de modèle est obtenue par l'identification. Ces modèles n'ont pas de significations physiques en soit mais ils s'attachent uniquement à décrire le comportement dynamique grâce à une formulation mathématique. On distingue deux familles de modèles de comportements :

- Les modèles de représentation ***non paramétriques*** : ces modèles sont principalement représentés par des courbes qui ne peuvent pas être décrites par un ensemble fini de nombres (exemples : réponse fréquentielle, réponse indicielle, …).
- Les modèles de représentation ***paramétriques*** : ces modèles sont caractérisés par un ensemble fini de nombre (exemple : les coefficients de polynômes pour la représentation par fonction de transfert, …).

Les modèles paramétriques peuvent être différemment étudiés suivant leurs représentations.

Représentation par équation aux différences : processus ARMA

Pour un système ne présentant pas de retard pur, cette représentation est parfois appelée équation récurrente, reliant la sortie $y(t)$ à l'entrée $u(t)$. Le processus ARMA (AutoRegressive Moving Average) est défini par :

$$y(t) = \sum_{j=1}^{n_b+1} \beta_j u(t-j) - \sum_{j=1}^{n_a} \alpha_j y(t-j)$$
$$\Leftrightarrow y(t) = \sum_{j=1}^{n_b+1} b_{j-1} u(t-j) - \sum_{j=1}^{n_a} a_j y(t-j) \Big| \begin{matrix} \beta_j = b_{j-1} \\ \alpha_j = a_j \end{matrix}$$

[2.2. 01]

Si l'on raisonne en discret, l'échantillon de sortie est connu à partir des échantillons d'entrée et de sortie précédents :

$$y_k = \sum_{j=1}^{n_b+1} b_{j-1} u_{k-j} - \sum_{j=1}^{n_a} a_j y_{k-j}$$

[2.2. 02]

Ce type de représentation convient tout particulièrement à l'identification paramétrique.

Représentation par fonction de transfert

La fonction de transfert (continue et discrète) est aujourd'hui un outil essentiel pour l'automatique. Elle permet l'étude des systèmes asservis (performances) et la conception des commandes (rejet des perturbations, précision, stabilité, robustesse, etc…)

La fonction de transfert continue est définie pour les systèmes linéaires quelconques invariants de type SISO (Single Input Single Output) décrivant une équation différentielle suivant la transformée de Laplace (conditions initiales nulles).

$$a_0 y(t) + a_1 \frac{dy(t)}{dt} + \ldots + a_{n_a} \frac{d^{n_a} y(t)}{dt^{n_a}} = b_0 u(t) + b_1 \frac{du(t)}{dt} + \ldots + b_{n_b} \frac{d^{n_b} u(t)}{dt^{n_b}} \qquad n_a > n_b$$

[2.2. 03]

$$\overset{Laplace}{\rightarrow} \qquad G(s) = \frac{b_0 + b_1 s + \ldots + b_{n_b} s^{n_b}}{a_0 + a_1 s + \ldots + a_{n_a} s^{n_a}} = \frac{Y(p)}{U(p)}$$

[2.2. 04]

L'utilisation intéressante de la représentation par fonction de transfert sous Laplace réside dans l'étude et la résolution systématique des systèmes physiques régis par une équation différentielle (système SISO linéaire).

La fonction de transfert pour les systèmes discrets est obtenue grâce à la transformée en z de la fonction de transfert continue. Le passage continu discret est décrit par la relation qui suit :

$$H(z) = \left(1 - z^{-1}\right) Z\left\{ L^{-1}\left(\frac{G(s)}{s} \right) \right\} = \frac{Y(z)}{U(z)}$$

[2.2. 05]

On peut également retrouver cette fonction de transfert à partir de la représentation par équation aux différences [2.2.1. 01] :

$$\begin{aligned} &\overset{TZ}{\Rightarrow} Y(z) = \sum_{j=0}^{n_b} b_j U(z) z^{-j-1} - \sum_{j=1}^{n_a} a_j Y(z) z^{-j} \\ &\Leftrightarrow Y(z)\left(1 + \sum_{j=0}^{n_a} a_j z^{-j} \right) = \sum_{j=0}^{n_b} b_j U(z) z^{-j-1} \Big|_{a_0 = 1} \\ &\Leftrightarrow \frac{Y(z)}{U(z)} = z^{-1} \frac{\sum_{j=0}^{n_b} b_j z^{-j}}{\sum_{j=0}^{n_a} a_j z^{-j}} = z^{-1} \frac{B(z^{-1})}{A(z^{-1})} \end{aligned}$$

[2.2. 06]

Remarque : si l'on note q^{-1} l'opérateur retard, on obtient la représentation polynomiale de la fonction de transfert discrète:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1)$$

[2.2. 07]

Autres représentations intéressantes

Utilisation de la convolution discrète

L'intérêt de cette représentation se trouve dans la démarche pratique d'obtention d'une fonction de transfert à partir d'une réponse impulsionnelle.

Soient $\{f_i\}$ les coefficients de la réponse impulsionnelle. En supposant le système causal, on peut écrire en utilisant la convolution discrète :

$$y(t) = \sum_{i=-\infty}^{+\infty} f_i u(t-i) \overset{systeme\_causal}{\Rightarrow} y(t) = \sum_{i=0}^{+\infty} f_i u(t-i)$$

[2.2. 08]

En utilisant la transformée en z, on obtient la fonction de transfert :

$$y(t) = \sum_{i=-\infty}^{+\infty} f_i u(t-i) \overset{TZ}{\Rightarrow} Y(z) = \sum_{t=-\infty}^{+\infty}\left(\sum_{i=-\infty}^{+\infty} f_i u(t-i)\right) z^{-t}$$
$$\Leftrightarrow Y(z) = \left(\sum_{i=-\infty}^{+\infty} f_i z^{-i}\right)\left(\sum_{u=-\infty}^{+\infty} u(u) z^{-u}\right)\Bigg|_{t-i=u} = F(z)U(z)$$

[2.2. 09]

Ce qui donne :

$$F(z) = \sum_{i=-\infty}^{+\infty} f_i z^{-i} \overset{systeme\_causal}{\Rightarrow} F(z) = \sum_{i=0}^{+\infty} f_i z^{-i}$$

[2.2. 10]

Pour passer à un modèle incrémental, il suffit d'introduire l'opérateur $\Delta = 1 - q^{-1}$ et ainsi :

$$\Delta y(t) = \sum_{i=0}^{+\infty} f_i \Delta u(t-i)$$

[2.2. 11]

Représentation avec perturbations

En présence de perturbations, il est possible de donner une représentation polynomiale globale du système :

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + v(t)$$

[2.2. 12]

Avec $J(q^{-1})v(t) = e(t)$, $e(t)$ étant une perturbation stochastique ou déterministe. Par ailleurs on choisit très souvent $J(q^{-1}) = \Delta = 1 - q^{-1}$ ce qui permet d'introduire une action intégrale. On obtient ainsi le modèle CARIMA ou ARIMAX (Auto Regressive Integrated Moving Average with External inputs):

$$A(q^{-1})\Delta y(t) = B(q^{-1})\Delta u(t-1) + \Delta v(t) = B(q^{-1})\Delta u(t-1) + e(t)$$

[2.2. 13]

2.3. Choix pour la modélisation des boucles de régulation du projet Gaz

Les systèmes à gaz des Expériences du LHC sont tous soumis à des contraintes physiques : pressions, débits, températures, etc. Ces variables sont intimement liées aux caractéristiques des mixtures gazeuses et aux structures des systèmes (tuyauterie, actionneurs, etc…).

Un bon nombre de formulations en thermodynamique et en mécanique des fluides peuvent apporter des éléments de modélisation. Que ce soit de l'utilisation de l'équation de Bernoulli à l'application de l'équation des gaz parfaits, il existe une grande palette d'actions pour la mise en œuvre d'un modèle de connaissance. Ces approches sont effectivement intéressantes pour l'étude précise des phénomènes à quantifier, à évaluer et à comprendre.

Dans les systèmes à gaz, la moindre variation d'une variable de comportement du système peut avoir une répercussion directe sur le comportement global. Par exemple, une action sur le débit en entrée peut avoir une influence sur la pression sur le buffer en sortie du détecteur. Dans ces circonstances la modélisation via les lois physico-chimiques complexifient considérablement l'étude des asservissements.

Les modèles de connaissance nécessitent aussi d'avoir accès à des variables mesurables et mesurées. Les installations à gaz ne sont hélas pas toujours pourvues de l'ensemble des capteurs nécessaires pour satisfaire cette condition.

Pour simplifier l'étude des boucles de régulations des systèmes à gaz, il est donc préférable d'utiliser des modèles de comportement qui sont plus efficacement exploitables pour l'établissement des lois de commande.

Les méthodes graphiques et paramétriques d'identification sont ainsi choisies.

3. L'identification du modèle pour le projet Gaz

L'identification du modèle de comportement a pour but de déterminer les paramètres du modèle. A partir de mesures expérimentales, l'objectif est d'obtenir un modèle ayant un comportement le plus similaire au comportement du procédé identifié.

Les systèmes à gaz des Expériences du LHC ont la particularité de ne pas fonctionner dans des conditions extrêmes (exemple : les très très basses températures en cryogénie). De ce fait, les non linéarités sont peu dépendantes des phénomènes physiques liés aux conditions de fonctionnement. En contrepartie, les non linéarités sont plus en relation avec les propriétés des mixtures gazeuses.

Exemple : Influence des ratios de débit pour une vanne analogique

Soit Kv le coefficient de la vanne calculé avec la formule qui suit :

$$Kv = \frac{Q}{490}\sqrt{\frac{\rho_G * T_1}{\Delta P * P_2}} \qquad [3.\ 01]$$

Avec

Q = débit en m^3/h

T_1 = température en K

ρ_G = gravité en kg/m^3

(Caractéristique du débit de type 'Equal Percentage' pour la vanne).

Pour un mélange gazeux Néon/CO_2 dans les proportions 90/10 et a 30.0 degrés Celsius on peut dire que la gravité est égale à :

$$\rho_{gaz,90/10} = \frac{\rho_{gaz}}{\rho_{air}} = \frac{0.93842}{1.20474} = 0.779 \quad kg/m^3$$

Avec

ρ_{Gaz}=0.93842 kg/l, la masse volumique de la mixture

$V_m = RT/p = 24.36 \quad litres/mole$, le volume molaire de la mixture

$\rho_{air} = 1.293 * \frac{273}{273 + T(celsius)}$, la masse volumique de l'air.

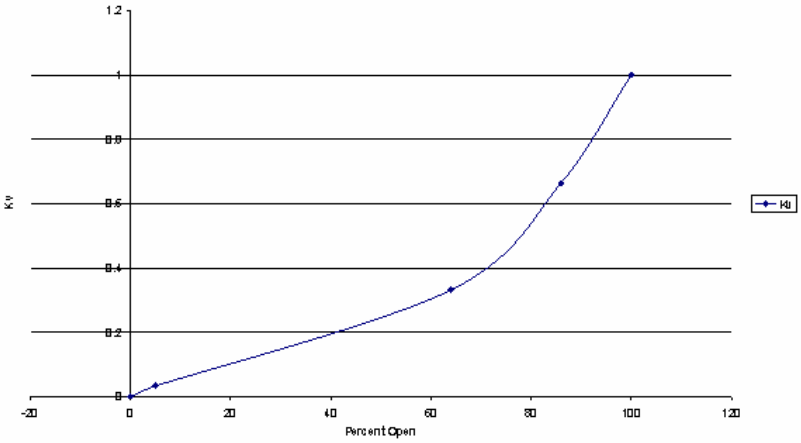
Soit une variation de 10 sur le ratio (proportion 80/20), la gravité devient :

$$G_{gaz,80/20} = 0.8616 \quad \text{kg/m}^3$$

Ainsi pour un débit constant de 15 Nm3/h (ex : ΔP=1.0, P2=0.5) on trouve :

$$Kv_{90/10} = 0.665\,,\ Kv_{80/20} = 0.699$$

Soit le relevé suivant pour une vanne de régulation régie par l'équation [3. 01] et ayant la caractéristique telle que :



Donc on peut voir que

$$Kv_{90/10} = 0.665 \quad \rightarrow \quad \approx 86\%$$
$$Kv_{80/20} = 0.665 \quad \rightarrow \quad \approx 87.5\%$$

En conclusion pour cet exemple, un changement de ratio de 90/10 à 80/20 entraîne une erreur sur l'ouverture de la vanne de l'ordre de 1.5% (soit donc une non-linéarité sur le comportement dynamique !).

~~~
~~~

Par la suite, on considère que les non linéarités sont assimilées à des erreurs de mesures ou à des perturbations du système dans l'hypothèse où les conditions de fonctionnement sont proches du nominal. Par ailleurs, l'identification des systèmes gazeux concerne les modèles de comportement linéaires mono-variables.

3.1. Les méthodes graphiques

Les méthodes graphiques sont très utiles pour la détermination rapide d'un modèle. La plupart des techniques existantes sont basées sur une réponse du procédé soumise à un échelon en entrée. De ce fait, ces méthodes sont souvent simples à mettre en œuvre sur les installations. Elles permettent d'obtenir des réponses sur le comportement de l'asservissement et ainsi de mieux appréhender le problème à résoudre. Bien que peu précises, les méthodes graphiques d'identification sont souvent suffisantes pour le réglage des contrôleurs simples tels que les PID.

Quatre techniques d'identification graphiques permettent d'obtenir des résultats intéressants :

- L'étude graphique dite intuitive pour les systèmes stables et instables
- La méthode de Broida pour les systèmes stables du premier ordre
- La méthode de Strejc pour les systèmes stables d'ordre n
- La méthode pour les systèmes oscillants

Remarque : avant de passer aux méthodes évoluées d'identification, les techniques graphiques sont très utiles dans les phases préliminaires.

Voir annexe E – Méthodes d'identification et de modélisation graphiques.

3.2. Les méthodes paramétriques

La régression linéaire est un outil puissant pour la résolution de problèmes d'estimation paramétrique. Cette technique permet de décrire le lien existant entre des variables contrôlées et des variables de réponse afin de prédire par la suite les réponses en fonction des variables contrôlées. La régression linéaire doit fournir l'équation de la droite s'ajustant au mieux aux points. La forme de l'équation est de la forme :

$$y(t) = \phi(t)\theta + \varepsilon(t)$$

[3.2. 01]

Où $y(t)$ est le vecteur des mesures, $\phi(t)$ la matrice d'observation, θ le vecteur des paramètres et $\varepsilon(t)$ le vecteur des erreurs.

Les méthodes se rattachant à ce principe permettent donc d'obtenir les paramètres d'un modèle dynamique linéaire.

L'identification paramétrique des systèmes dynamiques s'attache à trouver les paramètres en minimisant les erreurs entre modèles et procédés.

Ces méthodes s'y rattachant ont besoin de mesures expérimentales avec des signaux d'excitations de plus ou moins grandes amplitudes.

Le point clés de la mise en œuvre des algorithmes est l'élaboration d'un algorithme d'adaptation paramétrique (AAP) qui possède une structure récursive [Lan]. L'AAP est une approche élaborée basée sur la régression linéaire qui permet le calcul du nouveau vecteur des paramètres à partir de l'ancien estimé plus un terme de correction. Sa structure est telle que :

$$[A] = [B] + [C] \times [D] \times [E]$$

[3.2. 01]

A : Nouvelle estimation des paramètres (vecteur).
B : Estimation précédente des paramètres (vecteur).
C : gain d'adaptation (matrice).
D : Fonction des mesures (vecteur) ou « vecteur des observations ».
E : Fonction de l'erreur de prédiction (scalaire).

Soit le modèle discret tel que

$$y(t+1) = \theta^T \phi(t)$$

[3.2. 02]

Avec θ le vecteur des paramètres inconnus et $\phi(t)$ le vecteur des mesures

Soit le Prédicteur ***a priori,*** le modèle ajustable de prédiction construit sur la même structure que le modèle précédent :

$$\hat{y}^0(t+1) = \hat{y}(t+1|\hat{\theta}(t)) = \hat{\theta}(t)^T \phi(t)$$

[3.2. 03]

Avec $\hat{\theta}(t)$ le vecteur des paramètres estimés à l'instant t. L'erreur de prédiction a priori est ensuite définie par la relation:

$$\varepsilon^0(t+1) = y(t+1) - \hat{y}^0(t+1) = \varepsilon^0(t+1, \hat{\theta}(t))$$

[3.2. 04]

Enfin, l'objectif est de minimiser l'erreur de prédiction a priori grâce à l'AAP. La structure de l'AAP sera donc fonction des paramètres estimés à l'instant t, du vecteur de mesure et de l'erreur de prédiction a priori. L'AAP sera souvent de la forme :

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \Delta\hat{\theta}(t+1) = \hat{\theta}(t) + f\left(\hat{\theta}(t), \phi(t), \varepsilon^0(t+1)\right)$$

[3.2. 05]

3.2.1. Les méthodes intéressantes pour l'identification des systèmes Gaz

L'algorithme du Gradient

Cet algorithme est utilisé dans l'identification de modèle de type ARMA de la forme :

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t)$$

[3.2.1. 01]

L'erreur de prédiction étant définie telle que :

$$\varepsilon(t+1) = y(t+1) - \hat{y}(t+1)$$

[3.2.1. 02]

Et le critère de minimisation du gradient tel que :

$$\min_{\hat{\theta}(t+1)} J(t+1) = \left[\varepsilon(t+1)\right]^2$$

[3.2.1. 03]

L'APP est ensuite trouvé sous la forme :

$$\hat{\theta}(t+1) = \hat{\theta}(t) + \frac{F\phi(t)\varepsilon^0(t+1)}{1+\phi(t)^T F\phi(t)}$$

[3.2.1. 04]

Avec $\hat{\theta}(t+1)$ le nouveau vecteur des paramètres estimés à l'instant (t+1), $\phi(t)$ le vecteur des mesures et $F = \alpha I$ gain d'adaptation matriciel avec le poids α. Le poids doit satisfaire l'équation qui suit :

$$\alpha < \frac{2}{\phi(t)^T \phi(t)}$$

[3.2.1. 05]

L'algorithme des moindres carrés non récursifs (MC)

Cet algorithme est utilisé dans l'identification de modèle de type ARMA. On minimise le critère des « moindres carrés » tel que :

$$\min_{\hat{\theta}(t)} J(t) = \frac{1}{t}\sum_{i=1}^{t}[y(i) - \hat{\theta}(t)^T \phi(i-1)]^2 = \frac{1}{t}\sum_{i=1}^{t}\varepsilon^2(i, \hat{\theta}(t))$$

[3.2.1. 06]

On obtient le vecteur des paramètres sous la forme non récursive suivant :

$$\hat{\theta}(t) = \left[\sum_{i=1}^{t}\phi(i-1)\phi(i-1)^T\right]^{-1}.\sum_{i=1}^{t}y(i).\phi(i-1) = F(t)\sum_{i=1}^{t}y(i).\phi(i-1)$$

[3.2.1. 07]

Avec F(t):

$$F(t)^{-1} = \sum_{i=1}^{t}\phi(i-1)\phi(i-1)^T$$

[3.2.1. 08]

L'algorithme des moindres carrés récursifs (MCR)

Pour que l'algorithme MC soit récursif, on considère l'estimation de $\hat{\theta}(t)$ et $\hat{\theta}(t+1)$ tels que :

$$\begin{cases} \hat{\theta}(t) = F(t)\sum_{i=1}^{t} y(i).\phi(i-1) \\ \hat{\theta}(t+1) = F(t+1)\sum_{i=1}^{t+1} y(i).\phi(i-1) \end{cases}$$

[3.2.1. 09]

Et on remarque que :

$$F(t+1)^{-1} = F(t)^{-1} + \phi(t)\phi(t)^{T}$$

[3.2.1. 10]

On obtient la détermination du vecteur des paramètres sous la forme récursive grâce à l'APP qui suit :

$$\hat{\theta}(t+1) = \hat{\theta}(t) + F(t+1).\phi(t).\varepsilon^{0}(t+1)$$

[3.2.1. 11]

On définit ensuite le gain d'adaptation avec :

$$\begin{cases} F(t+1)^{-1} = \lambda_1(t)F(t)^{-1} + \lambda_2(t)\phi(t)\phi(t)^{T} \\ 0 < \lambda_1 \le 1 \\ 0 < \lambda_2 \le 2 \\ F(0) > 0 \end{cases}$$

[3.2.1. 12]

Et l'on trouve le gain d'adaptation :

$$F(t+1) = \frac{1}{\lambda_1(t)}\left[F(t) - \frac{F(t)\phi(t)\phi^{T}F(t)}{\frac{\lambda_1}{\lambda_2} + \phi(t)^{T}F(t)\phi(t)} \right]$$

[3.2.1. 13]

L'algorithme des moindres carrés récursifs (MCE)

Cet algorithme est utilisé dans l'identification de modèle de type ARMAX de la forme (modèle « procédé+perturbation ») :

$$A(q^{-1})y(t) = q^{-d}B(q^{-1})u(t) + C(q^{-1})e(t)$$

[3.2.1. 14]

Cette identification a pour objectif de modéliser le procédé ainsi que la perturbation afin d'obtenir une erreur de prédiction nulle. L'erreur de prédiction est telle que :

$$\varepsilon(t+1) = e(t+1) = y(t+1) - \hat{y}(t+1)$$

[3.2.1. 15]

On minimise le critère au sens des « moindres carrés » avec maintenant le vecteur des paramètres estimés et le vecteur des observations :

$$\phi(t)^T = \left[-y(t) \quad \ldots \quad -y(t-n_A+1) \quad u(t-d) \quad \ldots \quad u(t-d+n_B+1) \quad \varepsilon(t) \quad \ldots \quad \varepsilon(t-n_C+1)\right]$$
$$\hat{\theta}(t)^T = \left[\hat{a}_1(t) \quad \ldots \quad \hat{a}_{n_A}(t) \quad \hat{b}_1(t) \quad \ldots \quad \hat{b}_{n_B}(t) \quad \hat{c}_1(t) \quad \ldots \quad \hat{c}_{n_C}(t)\right]$$

[3.2.1. 16]

Et la convergence du vecteur des observations est sujette à une condition suffisante mais pas nécessaire :

$$\begin{cases} \dfrac{1}{C(z^{-1})} - \dfrac{\lambda_2}{2}; \\ 2 > \lambda_2 \geq \max \lambda_2(t) \end{cases}$$

[3.2.1. 17]

L'algorithme MVC : Maximum de Vraisemblance Récursif

Cet algorithme est une amélioration de la méthode des MCE. L'idée est de filtrer le vecteur d'observation par $1/\hat{C}(t,q^{-1})$ avec $\hat{C}(t,q^{-1})$ une estimation de $C(t)$ à l' instant *t*. L'effet est d'accélérer la décorrélation entre le vecteur des observations et l'erreur de prédiction. L'autre effet est de supprimer la condition sur $C(z^{-1})$ des MCE. Le vecteur des paramètres estimés et le vecteur des observations deviennent :

$$\phi(t)^T = \frac{1}{\hat{C}(t,q^{-1})} *$$
$$\left[-y(t) \quad \ldots \quad -y(t-n_A+1) \quad u(t-d) \quad \ldots \quad u(t-d+n_B+1) \quad \varepsilon(t) \quad \ldots \quad \varepsilon(t-n_C+1)\right]$$
$$\hat{\theta}(t)^T = \left[\hat{a}_1(t) \quad \ldots \quad \hat{a}_{n_A}(t) \quad \hat{b}_1(t) \quad \ldots \quad \hat{b}_{n_B}(t) \quad \hat{c}_1(t) \quad \ldots \quad \hat{c}_{n_C}(t)\right]$$

[3.2.1. 18]

3.3. Validation et estimation du modèle identifié

L'élaboration d'un modèle n'est en réalité qu'une étape du processus d'identification. Bien avant de modéliser le système, il est nécessaire de :

- Faire une acquisition correcte des mesures d'entrée/sortie. Cette étape s'accompagne quelquefois de traitements supplémentaires permettant une bonne estimation paramétrique (conditionnement des mesures).
- S'intéresser à la complexité du modèle choisi pour l'identification paramétrique (lorsque les méthodes graphiques sont limitées).
- Estimer les paramètres du modèle suivant une méthode d'identification.
- Valider le modèle trouvé.

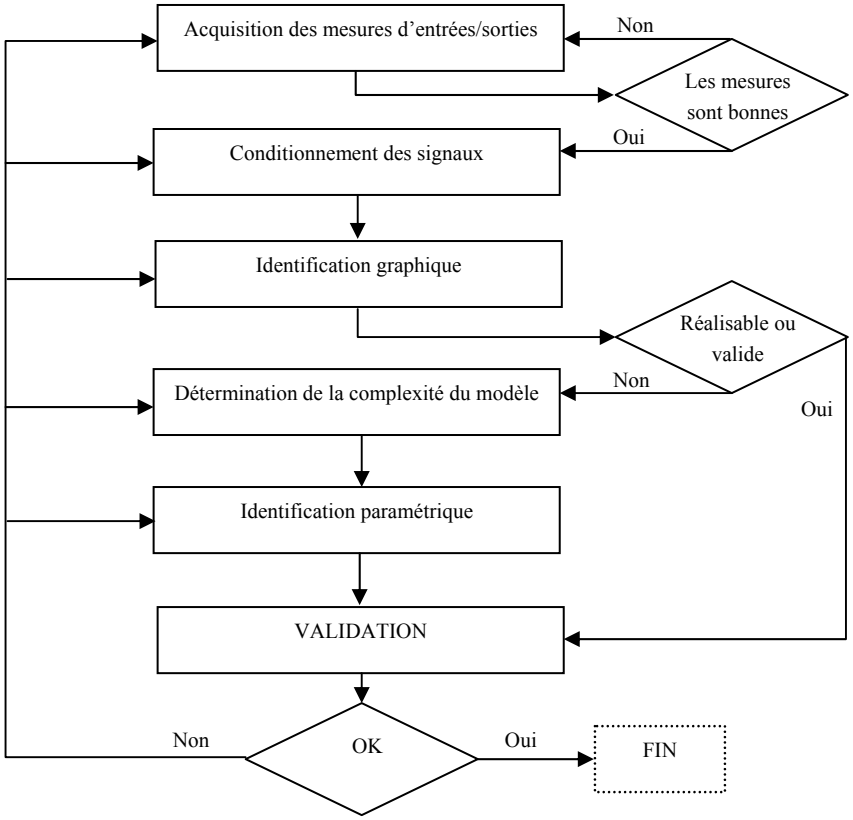


Figure 19 – Principe de l'identification process et de sa validation

La qualité des mesures est une condition nécessaire (mais non suffisante) pour l'obtention d'un bon modèle. Ajouté à cela, le conditionnement des signaux permet de s'affranchir de contraintes supplémentaires qui entravent l'identification.

Conditions initiales pour l'obtention des mesures :

- Les erreurs de précision et les erreurs de mesure doivent être négligeables (ou connues !).
- L'échantillonnage doit être fin pour une bonne résolution sans être trop exagérée (utilisation cohérente des ressources).
- Le système doit être dans les conditions nominales de fonctionnement.

- L'ajout de fonctions de filtrage est quelques fois nécessaire (filtre antirepliement par exemple).

Excitation du système :

- Il faut appliquer une entrée permettant de donner suffisamment d'information pour que l'identification puisse décrire la dynamique du système ainsi que le gain statique. Cette entrée doit être riche en fréquences et la solution théorique standard est d'appliquer une SBPA (séquence binaire pseudo aléatoire).
- En pratique la SBPA n'est pas toujours simple à générer. Une succession de trois échelons (suffisamment longs pour la détermination précise du gain) est souvent suffisante.

Conditionnement des mesures :

- Il faut enlever la composante continue (si elle est présente) résultante du point de fonctionnement. L'effet de celle-ci peut biaiser les estimations paramétriques pour la dynamique du système.
- Des valeurs aberrantes (autres que le bruit) doivent être enlevées (ou adaptées) pour ne pas corrompre l'identification. Celles-ci peuvent être dues aux instruments de mesures ou à un problème ponctuel des outils utilisés.
- Il peut être intéressant de changer les échelles des signaux pour que la vitesse de convergence lors de l'identification soit bonne.
- Lorsque le système possède un dérivateur pur ou un intégrateur pur, il est envisageable de prendre cette information en considération (par une opération sur les signaux) afin de réduire la complexité du modèle à identifier. Ce traitement peut améliorer la précision de l'identification.

Détermination de la complexité du modèle.

Le retard d est aisément déterminable pour les systèmes à gaz. A partir des mesures, on peut identifier le temps de réaction soumis à une excitation et ainsi déduire que d=(temps de réaction/échantillonnage).

L'estimation de l'ordre du système (ordre des polynômes A, B (C ?)) se fait à partir d'opérations sur les mesures. Pour ce faire, on peut utiliser le test des rapports des déterminants instrumentaux ou celui des résidus instrumentaux. Ces tests consistent en l'exploitation des couples de mesures entrée/sortie. On construit la ***matrice d'information*** Q_m telle que :

$$Q_m = \frac{1}{N}\sum_{k=1}^{N}\begin{bmatrix} u(k) \\ u(k+1) \\ u(k-1) \\ u(k+2) \\ .. \\ u(k-m+1) \\ u(k+m) \end{bmatrix}\begin{bmatrix} y(k+1) & u(k+1) & .. & y(k+m) & u(k+m) \end{bmatrix}$$

[3.3. 01]

Où m est l'ordre et N le nombre de mesures. Le test consiste à calculer **le rapport des déterminants instrumentaux** (RDI) tel que :

$$RDI(m) = \left| \frac{\det(Q_m)}{\det(Q_{m+1})} \right|$$

[3.3. 02]

Ensuite on retient comme ordre n du procédé la valeur m pour laquelle ce rapport augmente rapidement pour la première fois.

Le test des résidus instrumentaux (RI) correspond à calculer le rapport :

$$RI(m) = \left| \frac{\det(Q'_{m+1})}{\det(Q_m)} \right|$$

[3.3. 03]

Avec Q'_{m+1} la matrice obtenue à partir de la matrice Q_{m+1} en retirant la dernière ligne et la dernière colonne. L'ordre n du procédé est la valeur m lorsque le rapport RI diminue fortement

Ensuite de façon pratique on choisit généralement (si d est déterminé):

$$\begin{aligned} n_a &= n \\ n_b &= n_a \\ n_c &= n_a \end{aligned}$$

[3.3. 04]

Remarque : Il est aussi possible d'entreprendre l'estimation de la complexité en fin d'identification (sans les tests) en effectuant des approches successives sur le modèle et en déterminant sa pertinence par la validation.

Validation du modèle

La validation du modèle peut s'effectuer grâce à divers approches complémentaires :

- La première est de soumettre le modèle identifié à l'entrée des mesures pour visualiser la sortie et la comparer à la mesure réelle (simulation graphique par exemple). Par ce principe simple, on peut d'un seul coup d'œil dire si le modèle est faux. A contrario, il n'est pas évident d'affirmer par cette simple étape de la pertinence du modèle.
- La deuxième approche est de soumettre le modèle et le système réel (dans les conditions nominales) aux mêmes excitations afin de visualiser si l'identification était suffisamment précise pour décrire le comportement dynamique complet du système réel. Cette étape n'est hélas pas toujours aisée à réaliser.
- La troisième approche est de vérifier la validité du modèle par des tests sur les mesures. L'un de ceux-ci est basé sur le blanchissement de l'erreur de prédiction (méthode utilisées pour les MCR, MCE, MVR). On effectue le calcul tel que :

$$RN(i) = \frac{R(i)}{R(0)} = \left. \frac{\frac{1}{N}\sum_{t=1}^{N} \varepsilon(t)\varepsilon(t-i)}{\frac{1}{N}\sum_{t=1}^{N} \varepsilon^2(t)} \right|_{i=1,2,\ldots \max(n_a, n_b+d)}$$

[3.3. 05]

(N le nombre de mesures.)

En général un modèle est acceptable du point de vue du test si :

$$\left| RN(i) \leq \frac{2.17}{\sqrt{N}} \right|$$

[3.3. 06]

4. Implémentation et outils d'analyse pour les systèmes à gaz

Cette partie expose les outils d'analyses que j'ai développés sous automate programmables de marque Schneider avec Unity (choix du projet Gaz). Elle explique les solutions que j'ai proposées pour assurer la mise en œuvre pratique d'une démarche d'identification sous l'automate sans recours à Matlab/Simulink.

Voir annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB.

4.1. L'acquisition des données

L'acquisition des mesures se fait en temps réel dans l'automate.

Il peut être également utile de faire des relevés (enregistrements) en parallèle pour le travail sous Matlab au cas où des vérifications additionnelles sont requises. Dans ce cas, l'acquisition des données sous un protocole d'expérimentation fiable est assurée par l'outil ***DataStore*** de Schneider. Cet outil permet de faire des relevés en temps réel en se connectant aux automates programmables Schneider en local ou à distance (voir figures). DataStore utilise le standard de communication (Modbus, Modbus TCP/IP) via OFS.

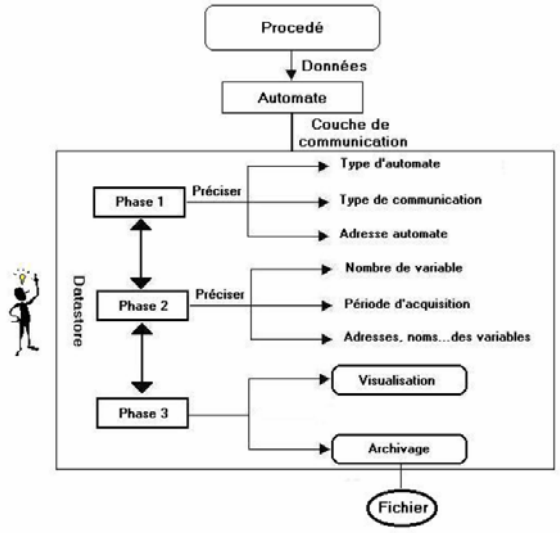


Figure 20 – Principe de fonctionnement de DataStore (source Schneider)

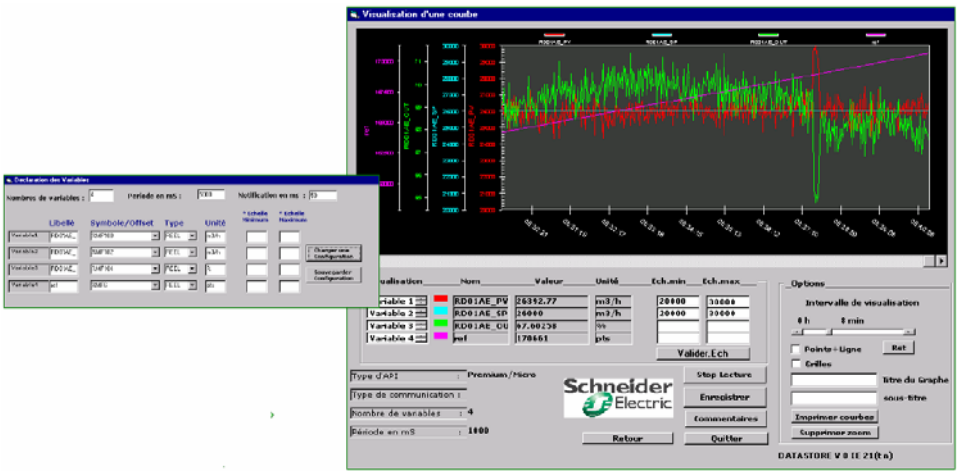


Figure 21 – Vue de DataStore en mode de lecture/enregistrement (source Schneider)

4.2. La détermination de la complexité du modèle

L'objet « RDI_sc » que j'ai implémenté sous UnityV2.1 permet de réaliser le test des RDI et RI en ligne. (Voir annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB.)

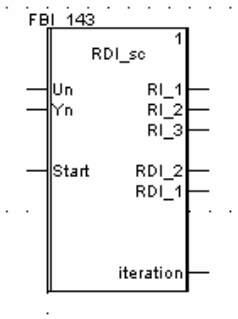


Figure 22 – Objet sous automate pour le test des RDI et RI

Le logiciel de développement pour automate permet de travailler avec des objets (DFB) munis d'attributs et de fonctions. UnityV2.1 est cependant limité à 7 encapsulations successives dans un même objet DFB. Cette restriction a pour effet de pouvoir déterminer le RDI jusqu'à i=2 et le RI jusqu'à i=3. La limitation est due aux tests des RDI et RI qui nécessitent de calculer les déterminants des matrices d'information Q_m $\left(Q_3|_{6\times 6}\right)$. Ainsi le test que je propose permet d'évaluer la complexité jusqu'à un troisième ordre.

Remarque : les processus industriels sont généralement d'ordre inférieur à trois. De ce fait le test RDI et RI sous UnityV2.1 peut être suffisant. Dans le cas où l'ordre est supérieur à cette limitation, le projet Gaz est obligé d'utiliser Matlab.

4.3. L'identification paramétrique

L'identification paramétrique est assurée grâce à trois objets DFB que j'ai développés pour le projet : le « MCR_SC » (Moindres Carrés Récursifs), le « MCE_SC » (Moindres carrés Etendus) et le « MVR_SC » (Maximum de Vraisemblance Récursif) (voir figure ci-dessous). (Voir annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB.)

Bien que ces trois stratégies d'identifications paramétriques ne soient pas les seules possibles, les trois objets que je propose suffisent à répondre à nombre de problématiques rencontrées pour le projet.

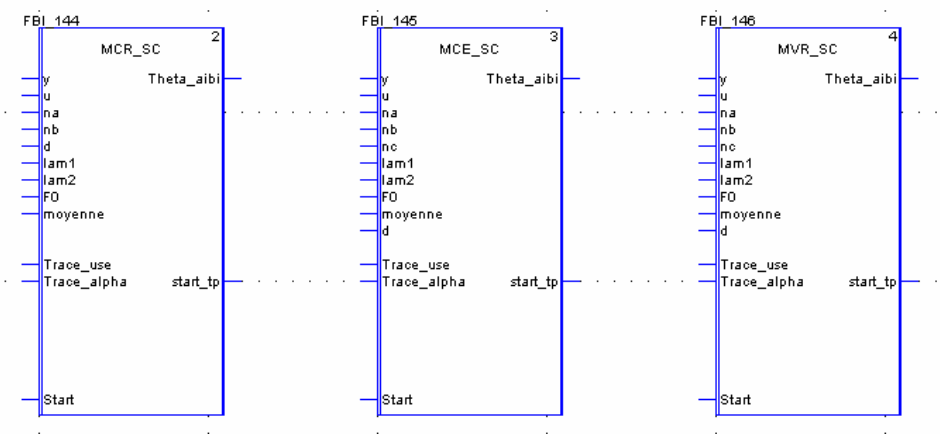


Figure 23 – Objets sous automate pour l'identification paramétrique sous UnityV2.1

4.4. Le signal d'excitation

L'utilisation d'un SBPA n'est pas nécessairement obligatoire pour assurer des mesures exploitables. Il s'avère que dans la pratique, trois échelons successifs (de U_{min} à U_{max}) suffisent pour obtenir une bonne excitation. L'objet « SBPAfix_sc » que j'ai implémenté réalise ce comportement (il faut juste spécifier le temps de stabilisation et les limites). Il permet aussi de faire un relevé des commandes $u_{\_moyennes}$ à soustraire pour l'identification. (Voir annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB.)

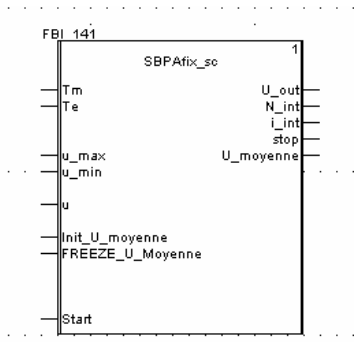


Figure 24 – Objet sous automate pour la génération de trois échelons successifs pour l'excitation du système

4.5. La validation du modèle

La validation du modèle identifié est assurée par deux objets DFB que j'ai développé à cet effet. (Voir annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB.)

- Le premier que j'ai appelé « SimuSysteme_sc » permet de simuler un système de type ARMA. Il peut également être utilisé pour construire un modèle CARIMA. Son utilisation permet de faire la comparaison processus/modèle dans l'automate pour la première phase de validation.
- Le deuxième que j'ai appelé « RNtype1_sc » réalise le test de blanchiment pour les trois méthodes d'identification développées pour le projet Gaz. Son utilisation est souvent combinée avec le premier objet.

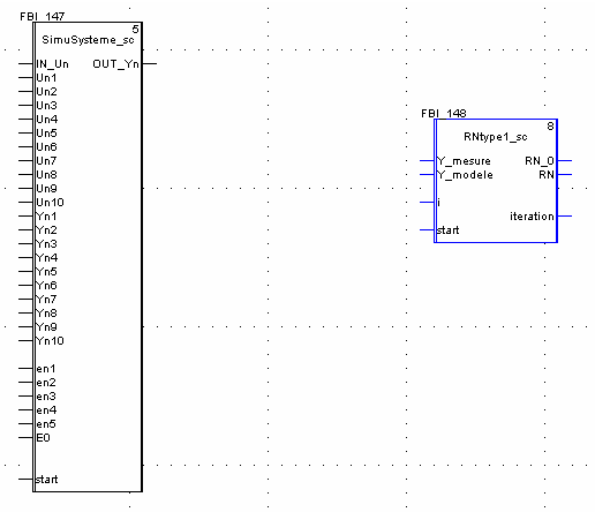


Figure 25 – Objets sous automate pour la validation du modèle identifié

5. Application : résultats d'identification en ligne sous automates pour la modélisation et l'identification de la pression Détecteur d'Alice TPC.

Alice TPC (Time Projection Chamber) est un sous-détecteur de l'Expérience ALICE. Sa fonction est de suivre les traces et la vitesse des particules chargées. Composé d'une enceinte gazeuse, le système à gaz TPC d'ALICE fait partie des 23 installations de contrôle-commande du projet Gaz. Le retour du gaz de TPC doit être compressé dans le module Pump afin d'avoir un recyclage efficace du gaz pour la purification et l'absorption du CO_2. C'est également à cet endroit que la régulation de pression s'effectue. La pompe en elle-même ne régule pas la pression.

C'est le module Pompe qui s'occupe de cette régulation par l'intermédiaire de la vanne analogique by-pass PV4003. L'objectif de cette régulation de pression est d'obtenir une pression à l'intérieur du sous détecteur de l'ordre de 0.5 mbar.

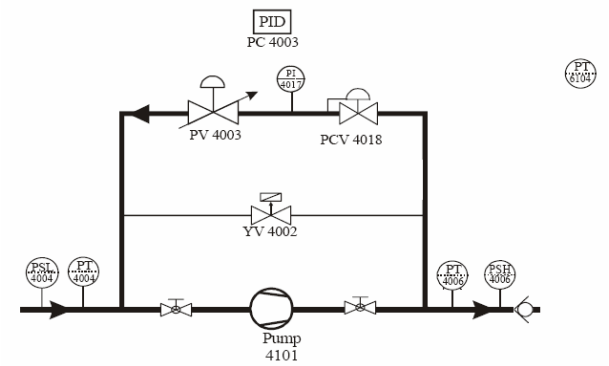


Figure 26 – Le module Pompe du sous détecteur ALICE TPC

Les résultats que j'ai trouvés et que j'expose sont de deux ordres :

- Montrer une démarche réelle d'identification en ligne d'une régulation spécifique d'un système à gaz.
- Comparer les résultats obtenus entre Unity (en ligne) et Matlab (hors ligne).

J'ai réalisé des mesures pour caractériser l'asservissement en pression. Ces mesures consistent en la manipulation de la vanne analogique PV4003 et de recueillir en sortie les variations de pression du détecteur. Pour travailler avec Matlab, j'ai fais l'acquisition des données via un programme d'acquisition (DataStore) développé par Schneider Electric (utilisant la technologie OPC, OFS V2.5).

J'ai conduit l'essai sur le détecteur lui-même. Les contraintes sont importantes car le risque d'endommager le détecteur est réel. Les mesures ont donc été effectuées en présence de l'expert gaz et dans les limites autorisées pour la pression [0 mbar, 1 mbar].

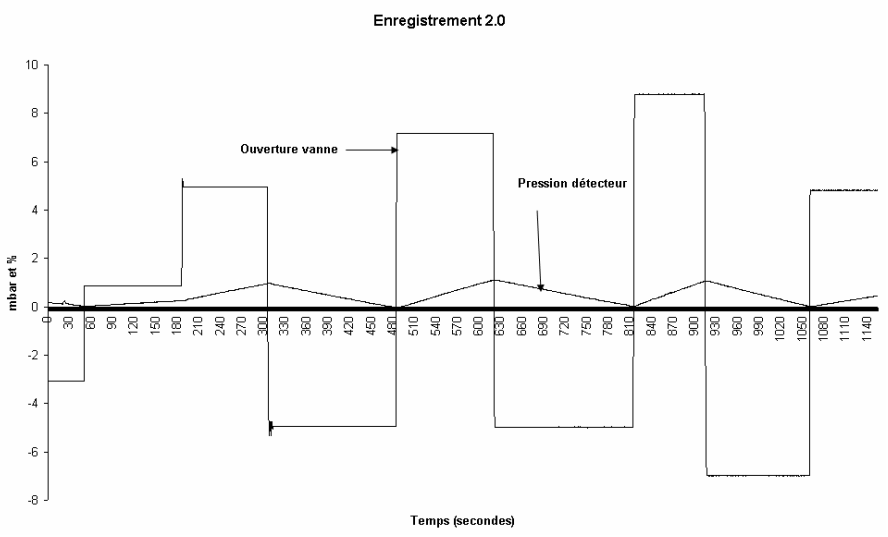


Figure 27 – Mesures pression Détecteur ALICE TPC

Le volume du détecteur TPC est de 90 m^3. La chambre est remplie de CO_2 et le débit du gaz est de 4 m^3/h. La pompe marche en continue avec une vitesse de 34.8 Hz. La pression moyenne relevée durant les mesures est de 0.45 mbar avec un écart type de 0.31 mbar. L'ouverture moyenne de la vanne est de 82.13% avec un écart type de 5.61 %.

De ces mesures j'ai constaté que le système est instable. J'ai aussi mis en évidence que le système possède un point critique de fonctionnement (82.13%) autour duquel il diverge dans les directions opposées des limites autorisées. Au-dessus de ce point d'ouverture de la vanne PV4003, la pression s'accroît en divergeant. En dessous, la pression diminue en divergeant. On remarque également sur le terrain que les mesures sont conditionnées par la pompe. Plus la vitesse de la pompe diminue, plus le point de fonctionnement critique de la vanne diminue. Enfin le système possède un retard pur d'environ 4 secondes.

5.1. Détermination de la complexité du modèle

J'ai réalisé le test en ligne des RDI pour plus de 2000 itérations. Les figures suivantes montrent les résultats que j'ai obtenus sous Matlab et Unity :

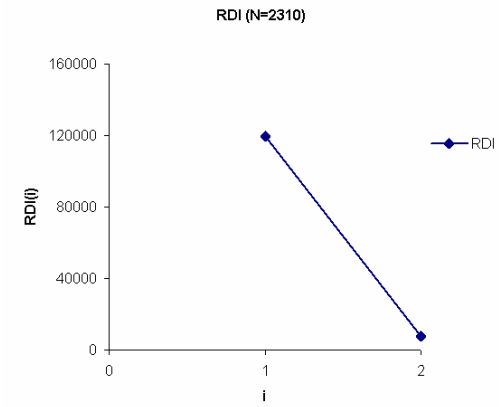


Figure 28 – Test des RDI sous Unity

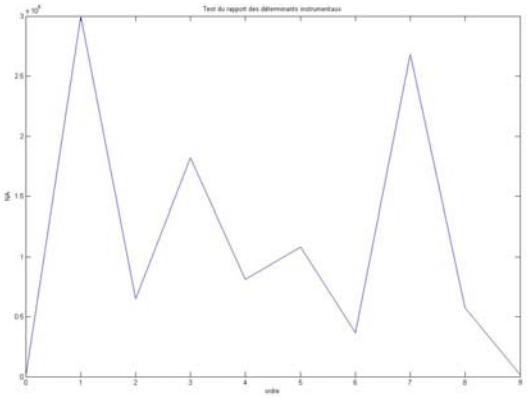


Figure 29 – Test des RDI sous Matlab

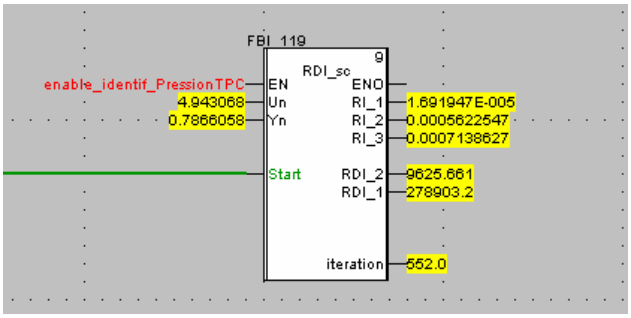


Figure 30 - Détermination en ligne du l'ordre du modèle en ligne sous Unity pour la pression détecteur d'Alice TPC

Ces résultats mettent en évidence l'ordre du modèle à identifier. On voit que n=1 et que donc $n_a = n_b = n_c = 1$.

5.2. Identification avec les MCR, MCE et MVR

Le modèle est un intégrateur d'ordre 1 avec un retard de 4 secondes. Ce premier résultat m'a permis d'orienter ma démarche d'identification en utilisant les trois méthodes récursives disponibles. Je trouve ainsi les modèles suivant avec Unity et Matlab :

$$y_{MCR\_Unity}(t) = \left(\frac{0.000342q^{-1}}{1-1.0043q^{-1}}q^{-8}\right)u(t)$$
$$y_{MCE\_Unity}(t) = \left(\frac{0.000567q^{-1}}{1-1.00031q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.256q^{-1}}{1-1.00031q^{-1}}\right)e(t)$$
$$y_{MVR\_Unity}(t) = \left(\frac{0.000066q^{-1}}{1-0.95448q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.063q^{-1}}{1-0.95448q^{-1}}\right)e(t)$$

[5.2. 01]

$$y_{MCR\_Matlab}(t) = \left(\frac{0.0005533q^{-1}}{1-0.9998q^{-1}}q^{-8}\right)u(t)$$
$$y_{MCE\_Matlab}(t) = \left(\frac{0.0005476q^{-1}}{1-0.9999q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.2411q^{-1}}{1-0.9999q^{-1}}\right)e(t)$$
$$y_{MVR\_Matlab}(t) = \left(\frac{0.0005654q^{-1}}{1-0.9476q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.0656q^{-1}}{1-0.9476q^{-1}}\right)e(t)$$

[5.2. 02]

Il est intéressant de constater que les identifications effectuées sous deux protocoles différents donnent des résultats proches les uns des autres. Les moindres carrés récursifs classiques semblent cependant obtenir une différence quant aux gains trouvés (paramètres b_1).

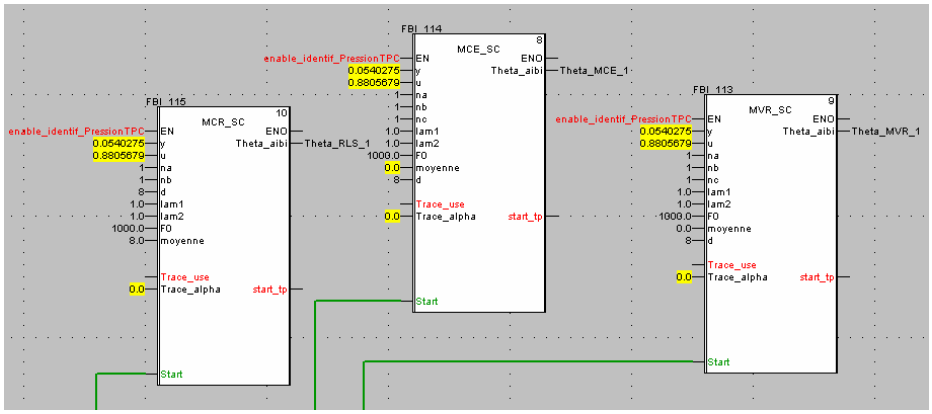


Figure 31 – Identification en ligne sous automate (MCR, MCE, MVR) pour la pression détecteur d'Alice TPC

Remarque : de ces identifications, la nature du système est stigmatisée: c'est un intégrateur. Pour éviter tout comportement divergeant (a_1<-1), il est préférable de prendre en compte cette information et de poser $a_1 = -1$. Ainsi les modèles peuvent être assimilés à :

$$y_{MCR\_Unity}(t) = \left(\frac{0.000342q^{-1}}{1-q^{-1}}q^{-8}\right)u(t)$$
$$y_{MCE\_Unity}(t) = \left(\frac{0.000567q^{-1}}{1-q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.256q^{-1}}{1-q^{-1}}\right)e(t)$$
$$y_{MVR\_Unity}(t) = \left(\frac{0.000066q^{-1}}{1-q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.063q^{-1}}{1-q^{-1}}\right)e(t)$$

[5.2. 03]

$$y_{MCR\_Matlab}(t) = \left(\frac{0.0005533q^{-1}}{1-q^{-1}}q^{-8}\right)u(t)$$
$$y_{MCE\_Matlab}(t) = \left(\frac{0.0005476q^{-1}}{1-q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.2411q^{-1}}{1-q^{-1}}\right)e(t)$$
$$y_{MVR\_Matlab}(t) = \left(\frac{0.0005654q^{-1}}{1-q^{-1}}q^{-8}\right)u(t) + \left(\frac{1-0.0656q^{-1}}{1-q^{-1}}\right)e(t)$$

[5.2. 04]

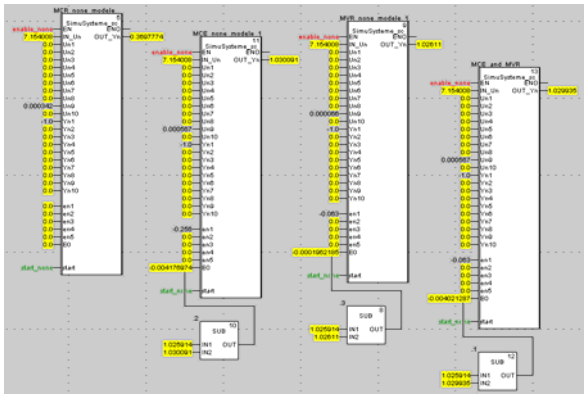


Figure 32 – Simulations des modèles sous automates pour la validation – pression détecteur d'Alice TPC

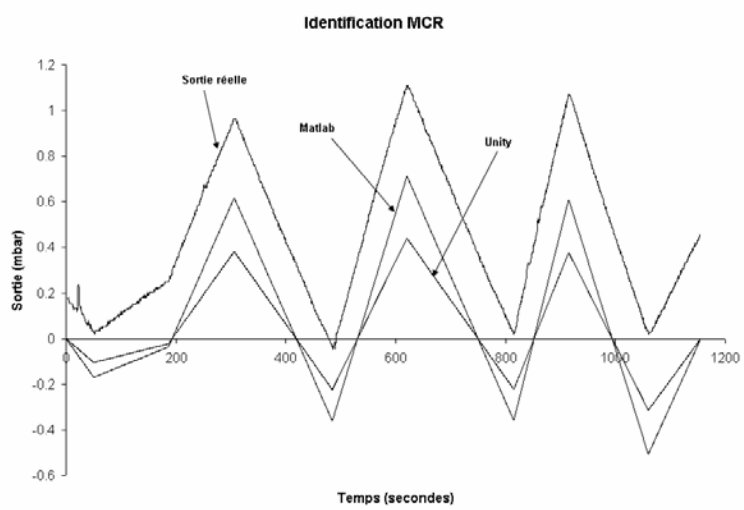


Figure 33 – Pression Détecteur ALICE TPC, Identification MCR

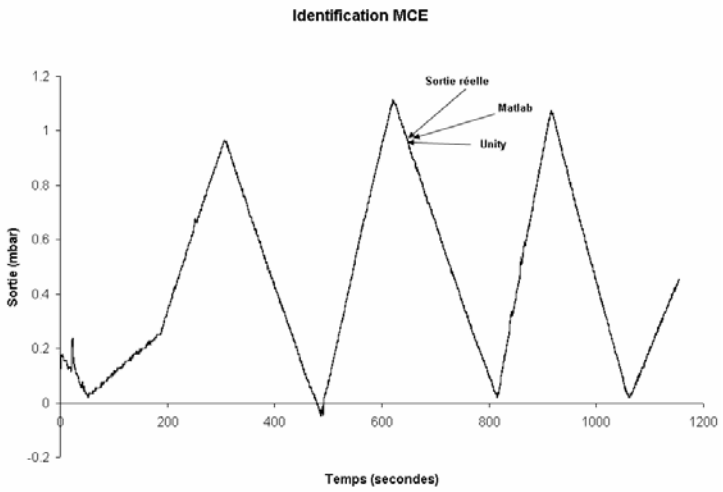


Figure 34 – Pression Détecteur ALICE TPC, Identification MCE

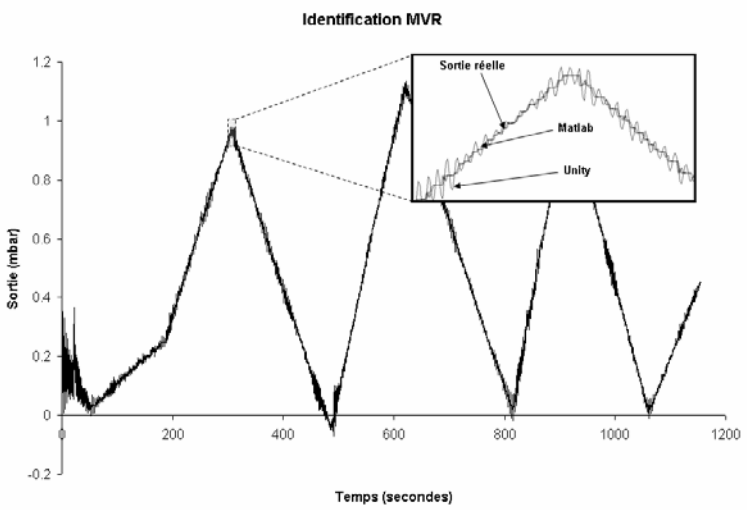


Figure 35 – Pression Détecteur ALICE TPC, Identification MVR

Les modèles que j'ai identifiés montrent que la méthode MCR n'est pas appropriée. Les méthodes MCE et MVR quant à elles donnent des résultats beaucoup plus satisfaisants aux vues des différences modèle-processus des figures précédentes.

Remarque : Il est important de comprendre que les identifications ne sont pas strictement identiques entre Unity et Matlab. Un point essentiel est à retenir : l'automate programmable tel qu'il est utilisé fonctionne avec un cycle d'exécution. Cela veut dire que la 'boucle' du programme qui s'exécute n'est pas synchrone mais dépend de la taille de l'application. De ce fait, en fixant un temps d'échantillonnage pour l'identification paramétrique, on est seulement sûr d'assurer un échantillon constant avec une erreur de plus ou moins un cycle d'exécution. L'identification n'étant pas strictement cadencée à une période d'échantillonnage précise, il est donc normal de voir des différences entre Unity et Matlab.

5.3. Validation

Les simulations des modèles obtenus par les identifications avec les MCE et MVR sont très satisfaisantes et montrent la difficulté à différencier les deux courbes. Le test de blanchiment sous Unity semble donc être indiqué pour l'aide à la décision et à la validation du modèle :

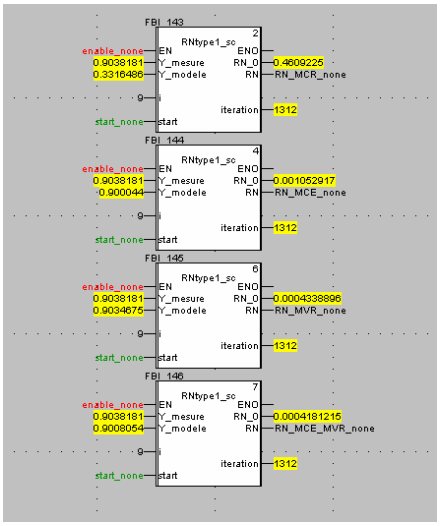


Figure 36 – Test de blancheur en ligne sous automates programmables – pression détecteur d'Alice TPC

	MCE	MVR
RN(0)	0.00107	0.000456
RN(1)	2.71E-05	5.43E-06
RN(2)	1.19E-05	1.50E-05
RN(3)	7.34E-06	1.30E-05
RN(4)	5.36E-08	1.24E-05
RN(5)	1.53E-06	1.52E-05
RN(6)	1.29E-06	1.52E-05
RN(7)	2.31E-06	1.31E-05
RN(8)	5.40E-06	1.95E-05

Figure 37 – Résultats du test de blancheur sous automate pour la pression détecteur d'Alice TPC

Le test de blanchiment de l'erreur de prédiction que j'ai appliqué met en évidence que les deux méthodes sont acceptables (car les covariances sont toutes en dessous de la limite théorique (0.0451)). Cependant ces résultats ne permettent pas de décider sur le modèle le plus pertinent à utiliser. Pour trancher sur ce point je me suis intéressé aux allures des convergences des paramètres pour les méthodes MCE et MVR.

Le paramètre a_1 étant connu (-1), je me suis uniquement intéressé aux évolutions de b_1 et c_1 :

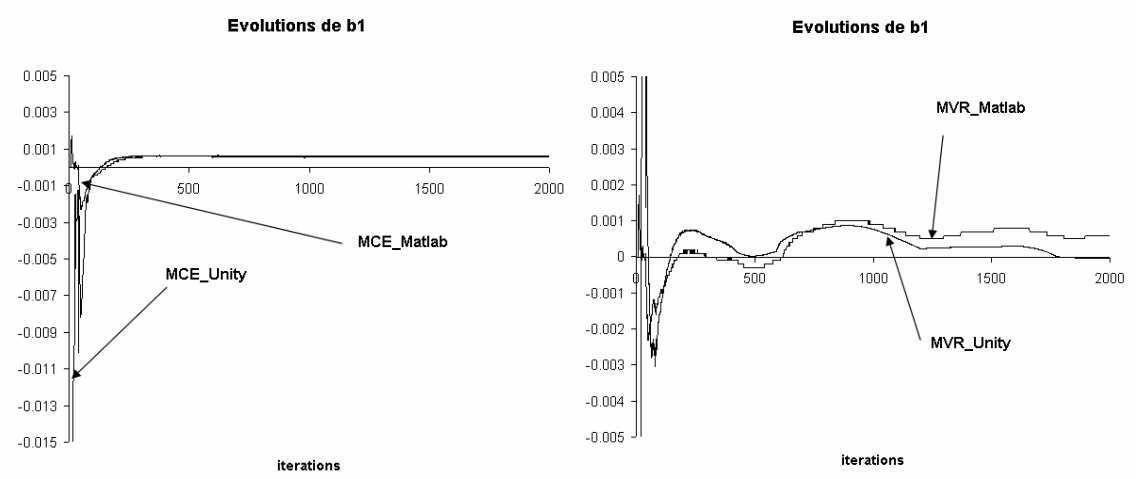


Figure 38 – Pression Détecteur ALICE TPC, Identification MCE et MVR, évolution de b_1

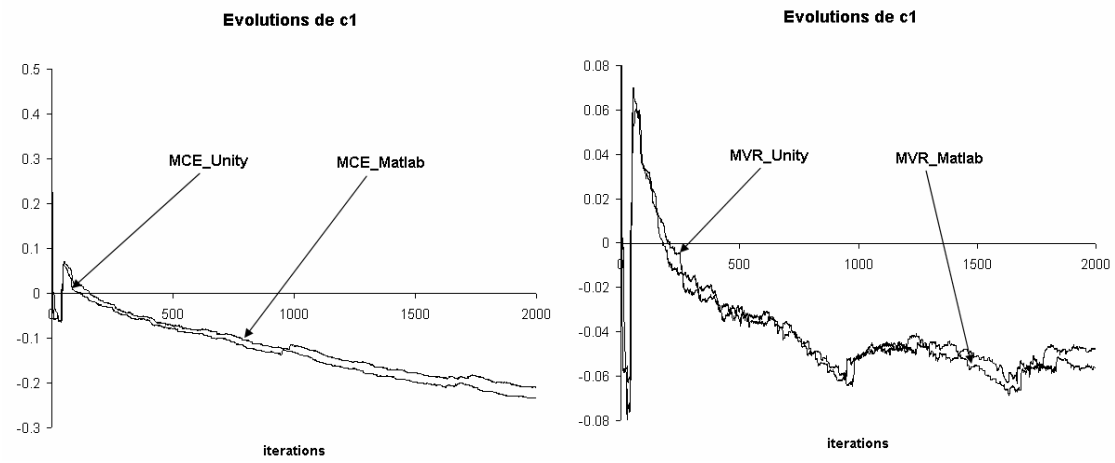


Figure 39 – Pression Détecteur ALICE TPC, Identification MCE et MVR, évolution de c_1

Les courbes précédentes enseignent plusieurs points :

- Les évolutions de b_1 et c_1 suivent des tendances proches de celles obtenues avec Matlab.
- La méthode des MCE montre une convergence rapide vers un paramètre b_1 stable.
- La méthode des MVR semble montrer une convergence relative pour le paramètre c_1.

5.4. Choix final

Mon choix s'est dirigé vers un coefficient b_1 identifié par la méthode des MCE et pour un coefficient c_1 identifié par la méthode MVR. Mon modèle pertinent est obtenu à partir de l'utilisation des MCE+MVR ; ce qui m'a permis d'élaborer un modèle ayant comme expression :

$$y_{MCE+MVR}(t) = \left(\frac{0.000567q^{-1}}{1-q^{-1}} q^{-8} \right) u(t) + \left(\frac{1-0.05q^{-1}}{1-q^{-1}} \right) e(t)$$

[5.3. 01]

6. Conclusion

Le processus d'identification/modélisation est une étape essentielle pour mener à bien la synthèse des boucles de régulations des systèmes à gaz. Le projet GCS utilise des automates programmables industriels de marque Schneider. L'utilisation de blocs fonctions dans l'automate permet d'introduire des algorithmes avancés réutilisables. Dans cette mouvance j'ai crée un ensemble comprenant plusieurs DFB pour faire de l'identification et de la modélisation en ligne. En dépits de quelques limitations inhérentes aux technologies de l'automate, les identifications sont pertinentes. Grace à un exemple concret sur site (pression du détecteur TPC d'ALICE), on constate que l'identification/modélisation est assurée pour donner des résultats proches de la simulation (Matlab).

CHAPITRE 3

CORRECTION DES SYSTEMES A GAZ

1. Le correcteur PID

2. Le Prédicteur de Smith

3. La commande prédictive fonctionnelle

4. La commande prédictive généralisée

5. Le régulateur RST

6. Implémentation des algorithmes avancés dans l'automate : résultats

7. Conclusion

Chapitre 3. Correction des systèmes à gaz

L'automate programmable industriel permet l'implémentation numérique d'algorithmes de contrôles pour assurer l'asservissement de grandeurs physiques clés pour le processus à gérer. Dans ce contexte, le calculateur (l'automate) assure une fonction de régulateur numérique pour ce qui à trait aux régulations. Les systèmes de contrôle à gaz du LHC ont donc la possibilité d'utiliser la puissance du numérique pour résoudre les problèmes de contrôle-commande.

Il est difficile d'énumérer l'ensemble des stratégies existantes pour corriger un système. Il subsiste ainsi des critères de sélection permettant de choisir des méthodes classiques ou avancées pour la synthèse de correcteurs numériques.

Les facteurs d'appréciation pour les systèmes à gaz sont :

- Une utilisation des solutions passées.
- Un temps de calcul raisonnable pour éviter toute charge excessive pour l'automate.
- Un paramétrage sans complexité débordante du contrôleur.
- Une implémentation orientée objet.
- L'utilisation des connaissances du processus.
- Une approche 'libre' de configuration d'un régulateur pour la spécification des performances.

Mes choix se sont portés vers les solutions suivantes :

- Le régulateur PID pour sa simplicité et sont utilisation connue.
- Le Prédicteur de Smith pour sa simplicité de réglage et son efficacité pour résoudre les problèmes de retard.
- Le Régulateur RST pour sa structure ouverte permettant la spécification des performances (placement de pôles, etc.)
- La commande prédictive fonctionnelle (PFC) pour la synthèse d'un correcteur robuste, son implémentation efficace et son réglage simple.
- La Commande Prédictive Généralisée (GPC) pour ses bonnes performances, sa robustesse, la mise en jeu raisonnable de paramètres et sa stratégie d'horizon fuyant.

1. Le correcteur PID

Le PID est sans conteste le régulateur le plus populaire et le plus utilisé. Son approche simple et rapide permet de résoudre nombres de problèmes de régulations. Les premiers PID commercialement disponibles sont apparus dans les années 30 avec dès les années 40, des études d'optimisation paramétriques (Ziegler). Dès lors leur utilisation n'aura cessé d'être constante dans l'industrie (PID analogique, pneumatique, électronique et numérique).

La popularité du contrôleur PID repose sur des constats simples : une utilisation répandue donc ayant fait ses preuves ; une simplicité relative de réglages (trois paramètres) ; une mise en œuvre nécessitant (en apparence) peu d'expérimentations.

Hans Eder [Ede] parle précisément du phénomène entourant le correcteur PID. Il explique que le contrôleur est majoritairement utilisé par les professionnels car ne requérant 'à tord' pas de travail préparatoire ni de tests préalables. Cette approche faussée peut hélas conduire à des erreurs de réglages dramatiques pour le process. Le plus étonnant est de constater que la plupart des utilisateurs n'ont pas plus de connaissances dans le réglage du régulateur PID qu'il y a 20 ou 30 ans.

La force de ce régulateur est donc qu'il combine trois actions élémentaires pour corriger le système : le correcteur proportionnel, le correcteur intégral et le correcteur dérivatif. Tous les trois ont le mérite d'apporter des actions concrètes sur l'écart mesuré sans avoir un lien (a priori) sur la connaissance du processus. La structure des PID ne possède aucune partie pouvant intégrer, à un certain niveau, une modélisation quelconque. C'est peut-être en ce sens que l'expérimentation peut sembler désuète alors qu'il n'en est rien.

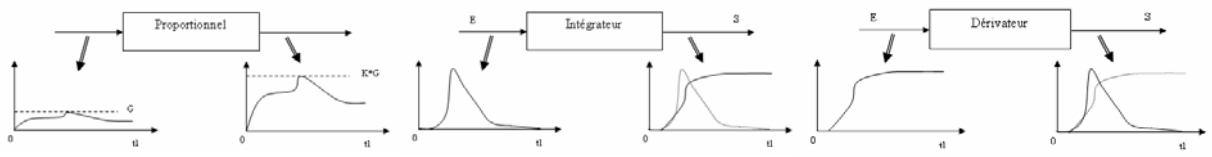


Figure 40 – Actions élémentaires du correcteur PID

Les méthodes de réglages de PID sont nombreuses [Pig] [JMF]. On peut citer :

- Le réglage par approches successives
- La méthode de Ziegler Nichols (temporelle et fréquentielle)
- La méthode de Takahashi.
- Les méthodes par optimisation de critères de Zhaung et Atherton [Zha].

2. Le Prédicteur de Smith

2.1. Historique

Le Prédicteur de Smith tient son nom de O.J. Smith qui proposa en 1959 une technique de compensation du retard du système par la mise au point d'un régulateur utilisant le PID traditionnel [Smi]. Cette solution consiste en l'élimination du retard pour le système en boucle fermée. De nombreuses méthodes appelées Prédicteur de Smith Modifié, Correcteur à Modèle Interne (IMC) ou Compensateur de Temps Mort (DTC), ont été développées depuis lors.

En 1968, A.T Fuller propose un contrôleur optimal non linéaire [Ful]. G. Alevisakis étend en 1973 la solution SISO du Prédicteur de Smith originel à une solution multi variable [Ale]. En 1977, J.F Donoghue fait une étude approfondie des méthodes mono et multi-variables et de l'approche optimale de la mise en œuvre du contrôleur [Don]. En 1981, K. Watanabe développe une structure modifiée du Prédicteur basée sur un PI ou un PID pour avoir un rejet de perturbation performant et une erreur statique nulle [Wat]. En 1994, K.J. Astrom propose une structure pour systèmes intégrateurs soumis à une perturbation [Ast]. M.R. Matausek trouve en 1996 une manière efficace de paramétrer un correcteur légèrement modifié proche de celui d'Astrom [Mat96].

2.2. Choix et principes pour le projet GCS

J'ai choisi pour le projet gaz, le Prédicteur originel de Smith pour les systèmes stables (du premier et second ordre) et le Prédicteur modifié de Matausek pour les systèmes intégrateurs. Ces choix sont le résultat de compromis techniques et de programmation permettant de couvrir le plus large éventail de cas possibles susceptibles d'être rencontrés (pour les systèmes à gaz).

Le principe du Prédicteur de Smith est simple. Il s'agit de trouver une structure telle que le retard pur n'intervienne plus dans l'équation caractéristique du système en boucle fermée.

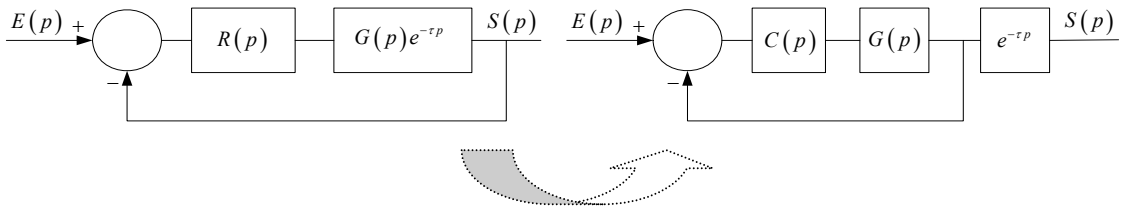


Figure 41 – Principe du Prédicteur de Smith

Dans chacun des cas nous avons :

$$S(p) = \frac{R(p)G(p)e^{-\tau p}}{1 + R(p)G(p)e^{-\tau p}} \qquad [2.2.\ 01]$$

$$S(p) = \frac{C(p)G(p)e^{-\tau p}}{1 + C(p)G(p)} \qquad [2.2.\ 02]$$

L'équivalence des deux systèmes se fait alors pour :

$$R(p) = \frac{C(p)}{1 + C(p)G(p)(1 - e^{-\tau p})} \qquad [2.2.\ 03]$$

Exemple du second ordre

Soit le procédé modélisé par un système du second ordre retardé G :

$$G(p) = \frac{G_s . e^{-\tau p}}{(1 + Tp)^2} \qquad [2.2.\ 04]$$

Le Schéma fonctionnel du Prédicteur est défini tel que :

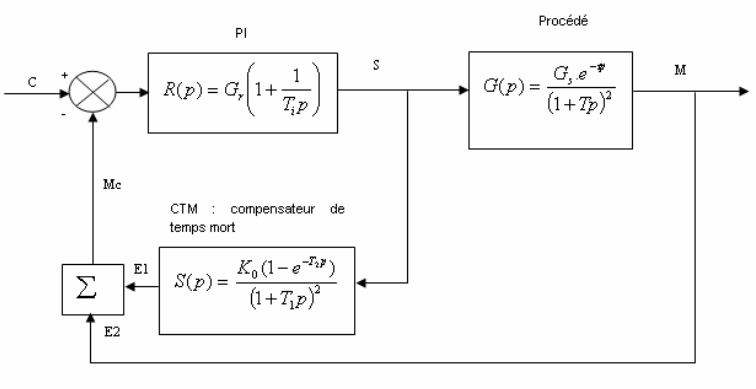


Figure 42 – Schéma fonctionnel du Prédicteur de Smith pour un second ordre stable

Le Prédicteur de Smith peut également être représenté avec le schéma ci dessous :

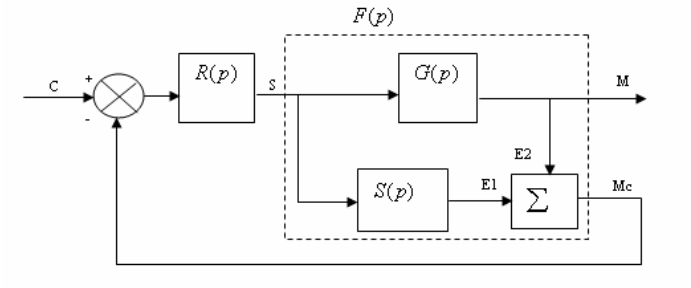


Figure 43 – Autre schéma fonctionnel du Prédicteur de Smith

F est donc la fonction de transfert vue par le correcteur PI tel que :

$$F(p) = G(p) + S(p) = \frac{G_s e^{-\tau p}}{(1+Tp)^2} + \frac{K_0(1-e^{-T_2 p})}{(1+T_1 p)^2} = \frac{G_s e^{-\tau p}}{(1+Tp)^2} + \frac{K_0}{(1+T_1 p)^2} - \frac{K_0 e^{-T_2 p}}{(1+T_1 p)^2}$$

$$\Leftrightarrow F(p) = \frac{G_s}{(1+Tp)^2}\Bigg|_{\substack{K_0=G_s \\ T_1=T \\ T_2=\tau}}$$

[2.2. 05]

Le retard est bien éliminé du point de vue du contrôleur PI ; ce qui nous donne en boucle fermée un second ordre :

$$BF = \frac{BO}{1+BO} = \frac{F(p).R(p)}{1+F(p).R(p)} = \frac{\frac{G_s G_r}{(1+Tp)^2}.\left(\frac{1+T_i p}{T_i p}\right)}{1+\frac{G_s}{(1+Tp)^2}.\left(\frac{1+T_i p}{T_i p}\right)} = \frac{\frac{G_s G_r}{(1+Tp)}.\frac{1}{Tp}}{1+\frac{G_s G_r}{(1+Tp)}.\frac{1}{Tp}}$$

$$\Leftrightarrow BF = \frac{G_s G_r}{(1+Tp)Tp + G_s G_r} = \frac{1}{1+\left(\frac{Tp+T^2p^2}{G_s G_r}\right)}$$

$$\Leftrightarrow BF = \frac{1}{1+\frac{T}{G_s G_r}p+\frac{T^2}{G_s G_r}p^2} = \frac{1}{1+ap+bp^2}$$

[2.2. 06]

Le facteur d'amortissement z (souvent ~ 0.7) de ce second ordre est donc définit tel que :

$$z = \frac{a}{2\sqrt{b}} = \frac{\frac{T}{G_s G_r}}{2\sqrt{\frac{T^2}{G_s G_r}}} = \frac{1}{2\sqrt{G_s G_r}}$$

$$\Rightarrow z^2 = \frac{1}{4.G_s G_r} \Leftrightarrow G_r = \frac{1}{4.G_s.z^2}$$

[2.2. 07]

Ce qui revient donc pour le réglage du Prédicteur de Smith pour un second ordre retardé :

$$S(p) = \frac{K_0(1-e^{-T_2 p})}{(1+T_1 p)^2} = \frac{G_s(1-e^{-\tau p})}{(1+Tp)^2}$$

$$R(p) = G_r\left(1+\frac{1}{T_i p}\right) = \frac{1}{4.G_s.z^2}\left(1+\frac{1}{Tp}\right)$$

[2.2. 08]

~~~

La méthode de Matausek consiste en la mise en œuvre d'un Prédicteur de Smith classique avec une modification sur la différence processus-modèle en vue de la compenser.

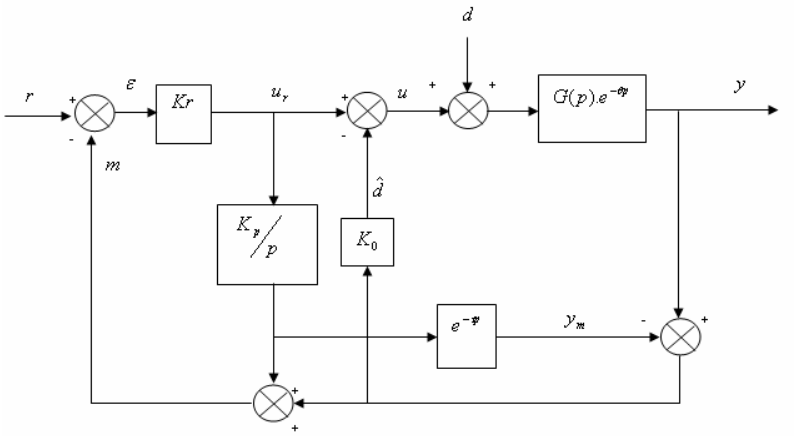


Figure 44 – Prédicteur de Smith de Matausek
~~~

Le système est défini tel que :

$$Y(p) = H_r(p).R(p) + H_d(p).D(s)$$

[2.2. 09]

Si l'on considère que « modèle = système » donc :

$$G(p) = \frac{K_p}{p}; \tau = \theta$$

[2.2. 10]

D'où :

$$H_r(p) = \frac{K_p.K_r.e^{-\tau p}}{1 + K_p.K_r}$$

[2.2. 11]

$$H_d(p) = \frac{K_p.\left[p + K_p K_r\left(1 - e^{-\tau p}\right)\right]e^{-\tau p}}{\left(p + K_p.K_r\right)\left(p + K_p.K_0.e^{-\tau p}\right)}$$

[2.2. 12]

De [2.2. 12] on peut voir que :

$$\lim_{p\to 0} H_d(p) = 0, \qquad K_0 \neq 0$$
$$\lim_{p\to 0} H_d(p) = \frac{1 + K_p K_r \tau}{K_r}, \qquad K_0 = 0$$

[2.2. 13]

Il faut donc trouver un K_0 non nul afin d'assurer un rejet des perturbations en régime stationnaire. Par ailleurs aux vues de la structure et de [2.2. 13] on peut dire que :

$$\lim_{t\to+\infty} \hat{d} = d, \qquad K_0 \neq 0$$

[2.2. 14]

Les tuning proposés par Matausek sont:

$$K_0 = \frac{\pi}{2.K_p.\tau}, \qquad Kr = \frac{1}{K_p.T_r}$$

[2.2. 15]

Avec T_r la constante de temps choisis pour le premier ordre en boucle fermée.

Grâce au Prédicteur de Smith (classique et modifié), les boucles de régulations ayant des temps morts conséquents peuvent être corrigées. Par une mise en œuvre simple et efficace le Prédicteur de Smith offre une fonctionnalité très intéressante pour le projet GCS.

3. La commande prédictive fonctionnelle

La commande prédictive fonctionnelle (PFC) a été introduite dans les années 80 par J. Richalet [Bou]. Elle fait partie des commandes prédictives dites à modèles internes (MBC). Le principe de cette commande se structure autour de quatre notions essentielles : le modèle, la trajectoire de référence, la commande structurée et la compensation de l'erreur modèle-processus (auto-compensateur) [Ric04] [Ric93].

3.1. Principes

En général, on considère le système à réguler comme un système modélisable suivant la représentation ARMA :

$$y_M(n) = \sum a_i . y_p(n-i) + \sum b_j . u(n-j)$$

Ensuite la PFC se base sur des principes qui lui sont propres pour sa mise en œuvre.

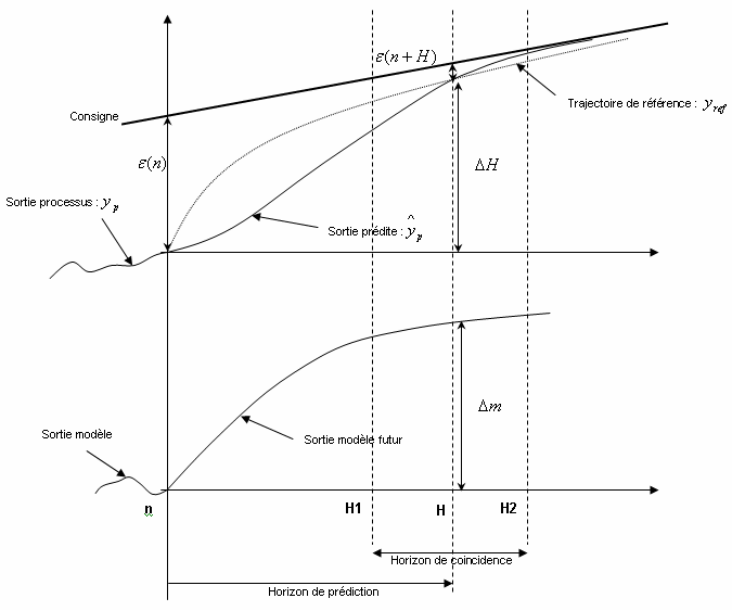


Figure 45 – Principe de la commande prédictive fonctionnelle (PFC)

Le premier principe intéressant de la commande prédictive fonctionnelle est l'utilisation d'une trajectoire de référence permettant de spécifier le comportement dynamique en boucle fermée. Généralement l'exponentielle est prise comme référence de telle sorte qu'à chaque échantillon passé, l'écart entre mesure et consigne doit être exponentiellement plus faible (voir figure). En notant ε(n) l'écart à l'instant n on a :

$$\varepsilon(n) = e^{-n T_e / T}$$
$$\varepsilon(n+1) = e^{-n T_e / T} . e^{-\frac{T_e}{T}} = \varepsilon(n).e^{-\frac{T_e}{T}} = \varepsilon(n).\lambda$$
$$\varepsilon(n+H) = \varepsilon(n).\lambda^H$$

[3.1. 01]

Le deuxième principe fondamental repose sur l'horizon de coïncidence (voir figure ci-dessus). Cet horizon est une spécification à la commande de faire coïncider la trajectoire de référence et la mesure. On parle volontiers de point de coïncidence si l'horizon se limite à un point.

Lorsque la dynamique voulue est spécifiée (horizon et trajectoire de référence), il est nécessaire de mettre en œuvre une commande (variable manipulée) assurant un comportement désiré.

La PFC introduit une notion bien particulière qui est de structurer la variable manipulée. Cela revient à choisir la commande future dans une base de fonctions prédéfinies, $U_{Bk}(i)$ soit :

$$MV(n+i) = \sum_{k=0}^{k_M} \mu_k U_{Bk}(i)$$

[3.1. 02]

Par superposition linéaire chaque fonction de base induit une sortie de base $y_{Bk}(i)$ du modèle (connue a priori, calculable ou tabulée pour un modèle donné) :

$$u_{Bk}(i) \rightarrow y_{Bk}(i)$$

[3.1. 03]

Il suffit ensuite de mettre en équation la commande. Pour illustrer cette démarche, on prend le choix que la commande est construite grâce à un point unique de coïncidence H. Ainsi on recherche l'incrément désiré $\Delta m = \Delta H$ de la sortie du processus à l'instant n+H (voir figure précédente). En H on peut dire que :

$\varepsilon(n) = C(n) - s_p(n) = C(n) - y_p(n)$ [Consigne – Sortie processus à l'instant n]

$\varepsilon(n+H) = C(n) - y_{reference}(n+H)$ [Au point de coïncidence H]

$\Delta H = y_{reference}(n+H) - y_p(n)$ [Objectif incrémentale donné au modèle]

On a donc :

$$\begin{aligned}
&\varepsilon(n+H) = C(n) - y_{reference}(n+H) = \varepsilon(n).\lambda^H = \left(C(n) + y_p(n)\right).\lambda^H \\
&\Leftrightarrow y_{reference}(n+H) = C(n) - \left(C(n) + y_p(n)\right).\lambda^H \\
&\Leftrightarrow y_{reference}(n+H) - y_p(n) = \left(C(n) - y_p(n)\right).\left(1 - \lambda^H\right)
\end{aligned}$$

D'où l'objectif à atteindre pour la commande:

$$\Delta H = \left(C(n) - y_p(n)\right).\left(1 - \lambda^H\right) \qquad [3.1.\ 04]$$

L'équation de commande au point H se résume enfin à :

$$\begin{aligned}
&\Delta H = \Delta m \\
&\Leftrightarrow \left(C(n) - y_p(n)\right).\left(1 - \lambda^H\right) = y_{Lachee,Modele}(n+H) + y_{Forcee,Modele}(n+H) - y_{Modele}(n)
\end{aligned} \qquad [3.1.\ 05]$$

Cette dernière expression est valide pour un point unique de coïncidence.

3.2. Résolution: exemples

3.2.1. Système du premier ordre

Soit un système régit par un premier ordre et représenté par son équation différentielle :

$$T\dot{y} + y = Ku \qquad [3.2.1.\ 01]$$

La solution à cette équation est :

$$y(t) = y_{lachee}(t) + y_{forcee}(t) = y_0.e^{-t/T} + K(1 - e^{-t/T}) \qquad [3.2.1.\ 02]$$

Il faut maintenant résoudre l'équation de commande [3.1. 05]. Pour réussir ce problème, il est intéressant de travailler sur le modèle discret du premier ordre et de travailler sur la variable manipulée par la suite. Soit donc le premier ordre de [3.2.1. 01] dans Laplace :

$$G(p) = \frac{S(p)}{E(p)} = \frac{K}{1 + Tp} \qquad [3.2.1.\ 03]$$

En discret on obtient l'équation en z :

$$G(z)=\frac{S(z)}{E(z)}=\frac{b.z^{-1}}{1+a.z^{-1}}=\frac{K(1-e^{-T_e/T}).z^{-1}}{1-e^{-T_e/T}.z^{-1}}$$
$$\Leftrightarrow S(z)(1-e^{-T_e/T}.z^{-1})=E(z).K(1-e^{-T_e/T}).z^{-1}$$
$$\Leftrightarrow S(n)=a_m S(n-1)+K(1-a_m)E(n-1)\big|_{a_m=e^{-T_e/T}}=S(n)_{Libre}+S(n)_{Forcée}$$

[3.2.1. 04]

Par récurrence et en considérant que la fonction de base est l'échelon : *MV(n)=MV(n+1)*, on peut déduire que :

$$S(n+1)=a_m S(n)+K(1-a_m)E(n)$$
$$\Leftrightarrow S(n+1)=a_m\left[a_m S(n-1)+K(1-a_m)E(n-1)\right]+K(1-a_m)E(n)$$
$$\Leftrightarrow S(n+1)=a_m^2 S(n-1)+Ka_m(1-a_m)E(n-1)+K(1-a_m)E(n)\big|_{E(n)=M(n)}$$

$$\Leftrightarrow S(n+1)=a_m^2 S(n-1)+K(1-a_m^2)MV(n)$$

[3.2.1. 05]

On peut en déduire que :

$$S(n+h)=a_m^h S(n)+K(1-a_m^h)MV(n)$$

[3.2.1. 06]

Dans l'équation de commande on obtient donc la variable manipulée *MV(n)* :

$$\Delta H=\left(C(n)-y_p(n)\right)\left(1-\lambda^H\right)=y_L(n+H)+y_F(n+H)-y_M(n)$$
$$\Leftrightarrow\left(C(n)-y_p(n)\right)\left(1-\lambda^H\right)=a_m^h S_M(n)+K(1-a_m^h)MV(n)-S_M(n)$$
$$MV(n)=\frac{\left(C(n)-y_p(n)\right)\left(1-\lambda^h\right)+S_M(n)-a_m^h S_M(n)}{K(1-a_m^h)}$$

[3.2.1. 07]

Pour la mise en œuvre de la PFC pour un système du premier ordre, la commande MV(n) est simple dans son expression. Son implémentation en est aussi simplifiée.

3.2.2. Application à un ordre quelconque à partir de sa réponse indicielle

Soit un système asymptotique stable défini par sa représentation de convolution sur N coefficients a_i. La réponse indicielle peut avoir une allure quelconque.

$$y_P(n) = a_1^P .u(n-1) + ... + a_i^P .u(n-i) + ... + a_N^P .u(n-N) = \sum_{i=1}^{N} a_i^P .u(n-i)$$

[3.2.2. 01]

(Ce système est du type MA (Moving Average) en temps réel $s_k = \sum_{i=0}^{p} a_i e_{k-i}$)

Pour déterminer la commande on suppose que la consigne varie en échelon et que la trajectoire de référence est l'exponentielle définie précédemment selon [3.1. 01]. La fonction de base est l'échelon, ce qui amène un seul coefficient μ_0 inconnu qui peut être déterminé par une coïncidence limitée en un seul point.

Soit donc le modèle associé :

$$y_M(n) = a_1^M .u(n-1) + ... + a_i^M .u(n-i) + ... + a_N^M .u(n-N)$$
$$\Leftrightarrow y_M(n) = \sum_{i=1}^{N} a_i^M .u(n-i)$$

[3.2.2. 02]

A l'instant n+H, le modèle devient :

$$y_M(n+H) = a_1^M .u(n-1+H) + ... + a_i^M .u(n-i+H) + ... + a_N^M .u(n-N+H)$$
$$\Leftrightarrow y_M(n+H) = a_1^M .u(n-1+H) + a_2^M .u(n-2+H) + ...$$
$$+ a_H^M .u(n-H+H) \quad + a_{H+1}^M .u(n-1) \quad + ...$$
$$+ a_{H+i}^M .u(n-i) \quad + ... \quad + \quad a_{H+i=N}^M .u(n-N+H)$$

$$\Leftrightarrow y_M(n+H) = a_1^M .u(n-1+H) + .. + a_H^M .u(n) + ... + a_N^M .u(n-N+H)$$

[3.2.2. 03]

Et comme pour le futur $u(n+i+H) = u(n), \forall i \in [1;H]$, on obtient donc le résultat ci-après pour le modèle :

$$y_M(n+H) = u(n).\sum_{i=1}^{H} a_i^M + \sum_{i=H+1}^{N} a_i .u(n-i+H)$$

[3.2.2. 04]

Si l'on note :

$$A_H^T = \left\{a_{H+1}^M, ...a_N^M\right\}$$
$$A^T = \left\{a_1^M, ...a_N^M\right\}$$
$$U^T(n) = \left\{u(n-1), u(n-2), ...u(n-N+H)\right\}$$

[3.2.2. 05]

L'équation de commande pour un système asymptotiquement stable défini par sa représentation de convolution s'obtient naturellement:

$$\Delta H = \left(C(n) - y_p(n)\right)\left(1-\lambda^H\right) = y_L(n+H) + y_F(n+H) - y_M(n)$$
$$\Leftrightarrow \left(C(n) - y_p(n)\right)\left(1-\lambda^H\right) = u(n).\sum_{i=1}^{H} a_i^M + A_H^T.U(n) - A^T.U(n)$$
$$\Leftrightarrow u(n) = \frac{\left(C(n) - y_p(n)\right)\left(1-\lambda^H\right) - A_H^T.U(n) + A^T.U(n)}{\sum_{i=1}^{H} a_i^M}$$

[3.2.2. 06]

A partir d'un simple relevé sur la réponse à un échelon du système (système quelconque auto stable), la PFC nous permet de calculer une commande avec un horizon et une trajectoire de référence choisis.

3.3. Réglage de la PFC

Le réglage de la PFC doit permettre d'assurer des performances acceptables. La méthode de Richalet est paramétrable grâce aux choix de la trajectoire de référence, de l'horizon de coïncidence et de la fonction de base. Le tableau qui suit résume les influences entre performances et paramètres :

	Précision	**Dynamique**	**Robustesse**
Fonction de base	2	0	0
Trajectoire de référence	0	2	1
Horizon de coïncidence	0	1	2

Figure 46 – Performances de la PFC et paramétrages du régulateur [Ric93]

4. La commande prédictive généralisée

La commande prédictive généralisée ou GPC (Generalized Predictive Control) est le nom employé par Clarke, Mohtadi et Tuffs dans leur formulation proposée de la commande prédictive à base de modèle [Cla87]. L'appellation GPC a été depuis largement adoptée comme une dénomination d'une classe de méthodes de contrôles prédictifs à caractères adaptifs [Cla].

La GPC est une structure complète qui s'assure de fournir un système corrigé stable pour un jeu de paramètres élaboré. Cette méthodologie permet de résoudre des problèmes de :

- processus à déphasage non minimal
- système intrinsèquement instable ou possédant des pôles mal amortis
- système avec temps morts variables ou inconnus
- système d'ordre inconnu

[Cla87]

Le principe de la commande prédictive réside dans la création d'un effet anticipatif. La mise en place de cette commande nécessite :

- l'élaboration d'un modèle du système à contrôler en vue de réaliser la prédiction du comportement futur de celui-ci.
- la mise en œuvre d'une séquence de commandes futures via l'utilisation d'un critère de minimisation (fonction de coût quadratique).
- Le renouvellement de la procédure à chaque itération. Seule la première commande est prise en compte et appliquée sur le système.

Voir annexe B – GPC – exemple d'une mise en œuvre pratique.

4.1. Principes de la mise en œuvre de la GPC

[Ram] [Bou]

Pour des raisons historiques, la mise en œuvre de la GPC s'effectue à partir du modèle représenté sous la forme CARMA (Controlled AutoRegressive Moving Average) :

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + x(t)$$

[4.1. 01]

Avec : *y(t)* sortie du processus, *u(t)* commande appliquée au système, q^{-1} opérateur retard, $x(t)$ terme lié aux perturbations, choisi habituellement sous la forme $x(t)=C(q^{-1})\xi(t)$ avec $\xi(t)$ séquence aléatoire non corrélée centrée.

Et les polynômes définis tels que :

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_n q^{-n} \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \ldots + b_m q^{-m} \\ C(q^{-1}) &= 1 + c_1 q^{-1} + \ldots + c_l q^{-l} \end{aligned}$$

[4.1. 02]

Le système devient :

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + C(q^{-1})\xi(t)$$

[4.1. 03]

On introduit une action intégrale $\Delta(q^{-1}) = 1 - q^{-1}$ pour éliminer l'erreur statique ; ainsi le modèle qui va être utilisé pour mettre en œuvre la séquence des commandes futures est :

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + C(q^{-1})\frac{\xi(t)}{\Delta(q^{-1})}$$

[4.1. 04]

Cette dernière équation est le modèle CARIMA (Controlled AutoRegressive Integrated Moving Average). Par la suite, on prend $C(q^{-1}) = 1$ en ne s'intéressant qu'aux fonctions de transfert entrée/sortie pour lesquelles ce polynôme n'a aucune influence.

La GPC utilise la notion de Prédicteur optimal qui a pour objectif d'anticiper le comportement du processus sur j-pas dans le futur sur un horizon fini.

A partir du modèle ([4.1. 04]), on élabore la sortie prédite :

$$y(t+j) = \underbrace{F_j(q^{-1})y(t) + H_j(q^{-1})\Delta u(t-1)}_{passé} + \underbrace{G_j(q^{-1})\Delta u(t+j-1) + J_j(q^{-1})\xi(t)}_{future}$$

[4.1. 05]

De l'équation [4.1. 04] on a :

$$A(q^{-1})\Delta(q^{-1})y(t) = B(q^{-1})\Delta u(t-1) + \xi(t) \qquad [4.1.\ 06]$$

A t+j on obtient :

$$A(q^{-1})\Delta(q^{-1})y(t+j) = B(q^{-1})\Delta u(t+j-1) + \xi(t) \qquad [4.1.\ 07]$$

De l'équation [4.1. 05] on peut écrire que :

$$\begin{aligned} y(t+j) &= q^{-j}F_j(q^{-1})y(t+j) + q^{-j}H_j(q^{-1})\Delta u(t+j-1) + \\ &\quad + G_j(q^{-1})\Delta u(t+j-1) + J_j(q^{-1})\xi(t+j) \\ \Leftrightarrow \left[1-q^{-j}F_j(q^{-1})\right]y(t+j) &= \left[G_j(q^{-1}) + q^{-j}H_j(q^{-1})\right]\Delta u(t+j-1) + \\ &\quad + J_j(q^{-1})\xi(t+j) \end{aligned} \qquad [4.1.\ 08]$$

De l'équation [4.1. 07], en multipliant par $J_j(q^{-1})$ on obtient alors :

$$A(q^{-1})J_j(q^{-1})\Delta(q^{-1})y(t+j) = B(q^{-1})J_j(q^{-1})\Delta u(t+j-1) + \xi(t)J_j(q^{-1}) \qquad [4.1.\ 09]$$

Ainsi de [4.1. 08] et [4.1. 09] on trouve les deux équations suivantes :

$$\begin{aligned} &A(q^{-1})J_j(q^{-1})\Delta(q^{-1}) = 1 - q^{-j}F_j(q^{-1}) \\ &\Leftrightarrow A(q^{-1})J_j(q^{-1})\Delta(q^{-1}) + q^{-j}F_j(q^{-1}) = 1 \end{aligned} \qquad [4.1.\ 10]$$

$$B(q^{-1})J_j(q^{-1}) = G_j(q^{-1}) + q^{-j}H_j(q^{-1}) \qquad [4.1.\ 11]$$

Ces équations sont également appelées équations diophantiennes.

En faisant l'hypothèse que la meilleure prédiction du terme lié aux perturbations est sa moyenne (ici nulle dans le cas du bruit blanc centré), le prédicteur optimal est défini de façon unique dès que les polynômes sont connus :

$$y(t+j) = F_j(q^{-1})y(t) + H_j(q^{-1})\Delta u(t-1) + G_j(q^{-1})\Delta u(t+j-1)$$

[4.1. 12]

$$\text{Avec : } \begin{cases} \deg\left[J_j(q^{-1})\right] = \deg\left[G_j(q^{-1})\right] = j-1 \\ \deg\left[F_j(q^{-1})\right] = \deg\left[A(q^{-1})\right] \\ \deg\left[H_j(q^{-1})\right] = \deg\left[B(q^{-1})\right] - 1 \end{cases}$$

[4.1. 13]

Pour construire la commande, la GPC doit mettre en place un critère de minimisation.

Pour ce faire la GPC décrit J qui est un critère de minimisation quadratique à horizon fini ; J est définis de la manière suivante :

$$J = \sum_{j=N_1}^{N_2} \left(y(t+j) - w(t+j)\right)^2 + \lambda \sum_{j=1}^{N_u} \Delta u(t+j-1)^2$$

[4.1. 14]

Avec : *w(t+j)* consigne à appliquer à l'instant (t+j), *y(t+j)* sortie prédite à l' instant (t+j), *Δu(t+j-1)* incrément de commande à l'instant (t+j-1), N_1 horizon de prédiction minimale sur la sortie, N_2 horizon de prédiction maximale sur la sortie, N_u horizon de prédiction sur la commande, λ coefficient de pondération sur la commande.

La minimisation du critère J fournit la séquence des commandes futures dont seule la première sera effectivement appliquée. Le coefficient λ donne plus ou moins de poids à la commande par rapport à la sortie, de façon à assurer la convergence lorsque le système de départ présente un risque d'instabilité.

4.2. Représentation matricielle et calcul de la commande optimale de la GPC

L'équation [4.1. 14] peut être représentée sous forme matricielle entre les horizons N_1 et N_2 telle que :

$$y = if(q^{-1}).y(t) + G.\tilde{u} + ih(q^{-1}).\Delta u(t-1)$$

[4.2. 01]

Avec :

$$if(q^{-1}) = \left[F_{N_1}(q^{-1}) \quad \ldots \quad F_{N2}(q^{-1})\right]^T$$

$$ih(q^{-1}) = \left[H_{N_1}(q^{-1}) \quad \ldots \quad H_{N2}(q^{-1})\right]^T$$

$$\tilde{u} = \left[\Delta u(t) \quad ... \quad \Delta u(t+N_u-1)\right]^T$$

$$G = \begin{bmatrix} g_{N_1}^{N_1} & g_{N_1-1}^{N_1} & \cdots & \cdots \\ g_{N_1+1}^{N_1+1} & g_{N_1}^{N_1+1} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ g_{N_2}^{N_2} & g_{N_2-1}^{N_2} & & g_{N_2-N_u+1}^{N_2} \end{bmatrix}$$

G est la matrice des coefficients $\left\{g_i^j\right\}$ des polynômes G_j ; $\left\{g_i^j\right\}$ correspondant aux valeurs des coefficients $\left\{g_i\right\}$ de la réponse indicielle du modèle.

Des relations [4.1. 14] et [4.2. 01] on peut réécrire le critère quadratique J tel que :

$$J = \left[if(q^{-1}).y(t) + G.\tilde{u} + ih(q^{-1}).\Delta u(t-1) - w\right]^T \left[if(q^{-1}).y(t) + G.\tilde{u} + ih(q^{-1}).\Delta u(t-1) - w\right] + \lambda.\tilde{u}^T\tilde{u}$$

[4.2. 02]

Avec : $w = \left[w(t+N_1) \quad ... \quad w(t+N_2)\right]^T$

La commande optimale s'obtient par la minimisation analytique du critère sous forme matricielle :

$$\frac{\partial J}{\partial u} = 0$$

[4.2. 03]

Or on a $\frac{\partial}{\partial u}\left[if(q^{-1}).y(t) + G.\tilde{u} + ih(q^{-1}).\Delta u(t-1) - w\right]^T = G^T$

D'où :

$$\frac{\partial J}{\partial u} = 2.G^T.\left[if(q^{-1}).y(t) + G.\tilde{u} + ih(q^{-1}).\Delta u(t-1) - w\right] + 2.\lambda.\tilde{u} = 0$$

$$\Leftrightarrow \tilde{u} = \left[G^T.G + \lambda.I_{N_u}\right]^{-1}.G^T\left[w - if(q^{-1}).y(t) - ih(q^{-1}).\Delta u(t-1)\right]$$

[4.2. 04]

On obtient ainsi la commande optimale :

$$\tilde{u} = M.\left[w - if(q^{-1}).y(t) - ih(q^{-1}).\Delta u(t-1)\right]$$

[4.2. 05]

Avec : $M = Q.G^T$ de dimension $N_u \times (N_2 - N_1 + 1)$

$Q = \left[G^T.G + \lambda.I_{N_u}\right]^{-1}$ de dimension $N_u \times N_u$

Enfin, seule la première valeur de la séquence [4.2. 05] est appliquée au système en accord avec la stratégie de l'horizon fuyant d'où :

$$\Delta u_{opt}(t) = m_1{}^T.\left[w - if(q^{-1}).y(t) - ih(q^{-1}).\Delta u(t-1)\right]$$

[4.2. 06]

Avec : m_1, première ligne de la matrice M.

5. Le régulateur RST

La forme polynomiale RST d'un contrôleur est une exploitation très intéressante des possibilités de la commande numérique. [Long06] [Bor93]

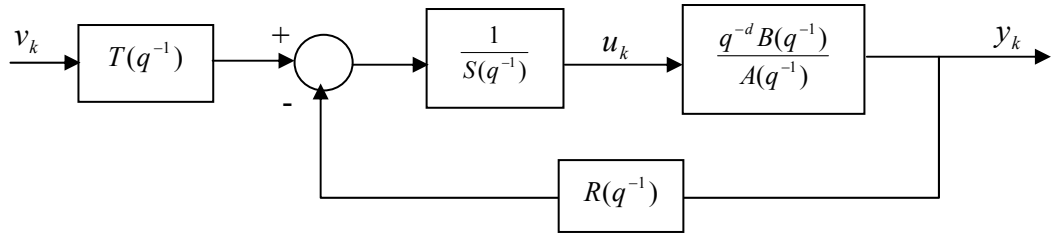


Figure 47 – Structure du régulateur RST

Le régulateur possède deux degrés de liberté : le premier classiquement défini sur le signal d'erreur consigne/mesure, le second autorisant la poursuite via une trajectoire de référence. La structure du contrôleur permet aussi d'imposer des pôles et certains zéros en boucle fermée.

5.1. Principe

Le procédé est donné par :

$$F(q^{-1}) = \left.\frac{q^{-d}B(q^{-1})}{A(q^{-1})}\right|_{\substack{\deg(B)=m \\ \deg(A)=n}}$$

[5.1. 01]

Et les polynômes du régulateur RST par :

$$T(q^{-1}) = t_0 + t_1 q^{-1} + t_2 q^{-2} + ...$$
$$R(q^{-1}) = r_0 + r_1 q^{-1} + r_2 q^{-2} + ...$$
$$S(q^{-1}) = s_0 + s_1 q^{-1} + s_2 q^{-2} + ...$$

[5.1. 02]

Le système en boucle ouverte s'écrit:

$$G(q^{-1}) = \frac{q^{-d}.T(q^{-1}).B(q^{-1}).R(q^{-1})}{S(q^{-1}).A(q^{-1})}$$

[5.1. 03]

Et le système en boucle fermée devient :

$$H(q^{-1}) = \frac{T(q^{-1}).F(q^{-1})/S(q^{-1})}{1 + R(q^{-1}).T(q^{-1}).F(q^{-1})/S(q^{-1})}$$
$$\Leftrightarrow H(q^{-1}) = \frac{q^{-d}.T(q^{-1}).B(q^{-1})}{S(q^{-1}).A(q^{-1}) + q^{-d}.R(q^{-1}).T(q^{-1}).B(q^{-1})}$$

[5.1. 04]

Soit en notant P le polynôme du dénominateur :

$$P(q^{-1}) = S(q^{-1}).A(q^{-1}) + q^{-d}.R(q^{-1}).T(q^{-1}).B(q^{-1})$$

[5.1. 05]

D'où :

$$H(q^{-1}) = \frac{q^{-d}.T(q^{-1}).B(q^{-1})}{P(q^{-1})}$$

[5.1. 06]

5.2. Le placement de pôle et la poursuite

Le choix de P permet d'imposer les pôles du système. Pour ce faire, il suffit de résoudre l'équation:

$$P(q^{-1}) = 1 + p_1 q^{-1} + p_2 q^{-2} + ... = S(q^{-1}).A(q^{-1}) + q^{-d}.R(q^{-1}).T(q^{-1}).B(q^{-1})$$

[5.2. 01]

La résolution de cette équation nécessite la compatibilité des degrés des polynômes. On appelle cette équation « identité de Bézout » ou « équation Diophantienne ».

Une autre possibilité offerte par la structure polynomiale du correcteur RST est la réalisation de la poursuite. L'idée est d'imposer une trajectoire de référence correspondant à un modèle de référence par la synthèse d'un régulateur RST. Ce qui revient à résoudre :

$$F_m(q^{-1}) = \frac{B(q^{-1})T(q^{-1})}{A(q^{-1})R(q^{-1}) + B(q^{-1})S(q^{-1})} = \frac{B_m(q^{-1})}{A_m(q^{-1})}$$

[5.2. 02]

5.3. Applications : représentation RST de la commande prédictive généralisée

Le correcteur RST possède une structure ouverte puissante pour représenter des commandes complexes. Par exemple la GPC peut être transcrite (et donc implémentée !) avec cette structure polynomiale.

De l'équation [4.2. 06] on peut écrire que :

$$\Delta u_{opt}(t)\left[1 + m_1{}^T.ih(q^{-1}).q^{-1}\right] = m_1{}^T.w - m_1{}^T.if(q^{-1}).y(t)$$

[5.3. 01]

Avec : $w = \left[w(t+N_1) \quad ... \quad w(t+N_2)\right]^T = \left[q^{N_1} \quad ... \quad q^{N_2}\right]^T w(t)$

D'où :

$$\Delta u_{opt}(t)\left[1 + m_1{}^T.ih(q^{-1}).q^{-1}\right] = m_1{}^T.\left[q^{N_1} \quad ... \quad q^{N_2}\right]^T w(t) - m_1{}^T.if(q^{-1}).y(t)$$

[5.3. 02]

La formulation RST du contrôleur GPC est donc définie tel que :

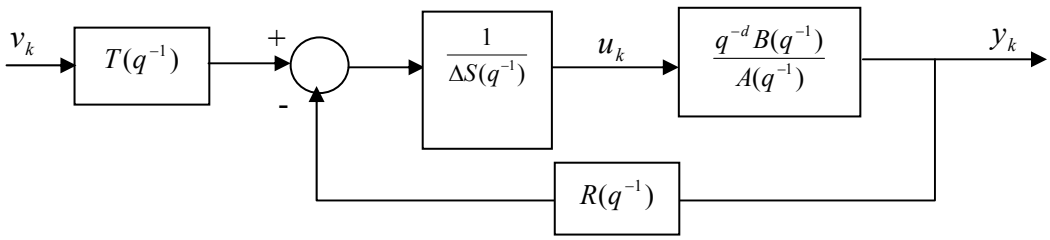


Figure 48 – Structure RST du contrôleur GPC

De cette structure on déduit que :

$$S(q^{-1}).\Delta(q^{-1}).u(t) = T(q^{-1}).w(t) - R(q^{-1}).y(t)$$

[5.3. 03]

Par identification entre [5.2. 02] et [5.2. 03], on obtient les expressions des polynômes de la structure RST :

$$\begin{cases} S(q^{-1}) = 1 + m_1^T.ih(q^{-1}).q^{-1} \\ R(q^{-1}) = m_1^T.if(q^{-1}) \\ T(q) = m_1^T.\left[q^{N_1} \quad \ldots \quad q^{N_2}\right]^T \end{cases}$$

[5.3 04]

Remarque : $T(q)$ renferme la structure non causale inhérente à la commande prédictive.

6. Implémentation des algorithmes avancés dans l'automate : résultats

L'objectif de cette partie est d'exposer des résultats concrets de corrections de systèmes divers grâce à des algorithmes que j'ai implémentés dans l'automate programmable Premium de Schneider. Le détail des objets DFB que j'ai réalisés se trouve dans l'annexe C.

6.1. Le Prédicteur de Smith

Le Prédicteur de Smith est un contrôleur très intéressant si le système à corriger possède un temps mort non négligeable. L'implémentation sous automate programmable industriel assure la correction des systèmes stables (premier et deuxième ordre) et des systèmes intégrateurs (instables).

Correction d'un système stable du premier ordre avec retard

Soit un système du premier ordre de la forme :

$$G(p) = \frac{2.0e^{-\tau p}}{1+10p}$$

[6.1. 01]

Les relevés des figures sont effectués pour des retards croissants (de 1 seconde à 12 secondes) et avec un temps d'échantillonnage de 500ms :

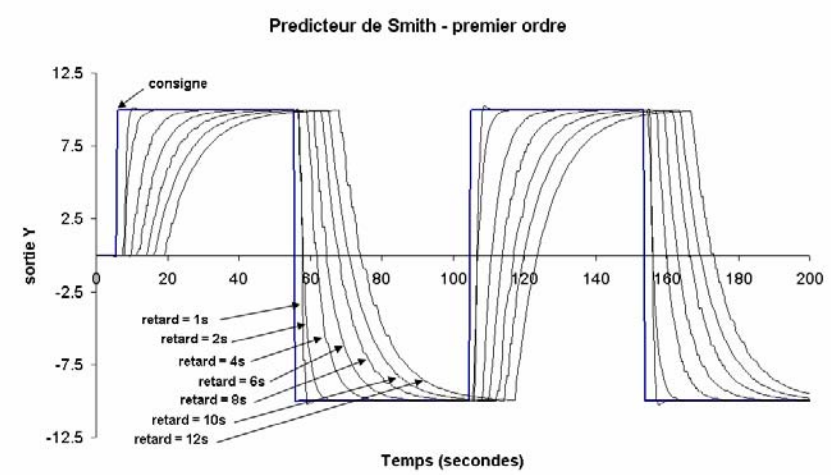


Figure 49 – Correction avec un Prédicteur de Smith d'un système stable du premier ordre avec retard – relevé 1 – sortie du système (mesure)

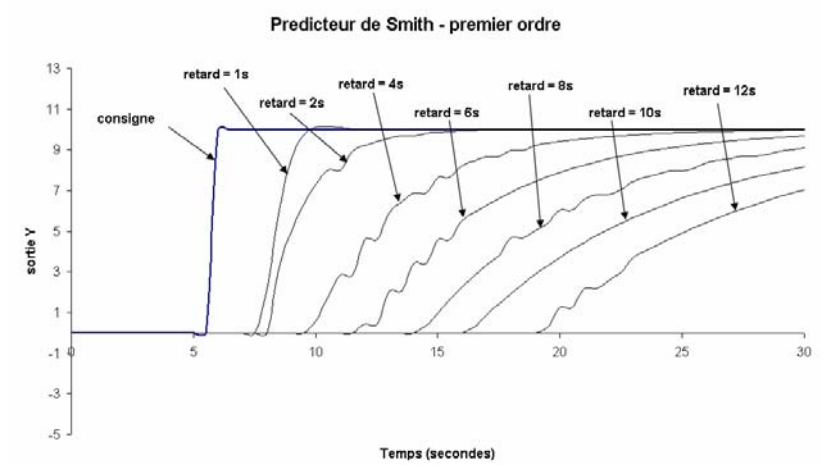


Figure 50 – Correction avec un Prédicteur de Smith d'un système stable du premier ordre avec retard – relevé 2 – sortie du système (mesure)

Les courbes précédentes mettent en avant la spécificité du Prédicteur de Smith : il permet d'annuler les effets classiques indésirables du retard. Les sorties du système résultantes des changements de consignes montrent la précision des réponses et la quasi inexistence de dépassement.

Les courbes qui suivent traduisent les commandes mises en œuvre pour la correction du premier ordre avec des retards variables :

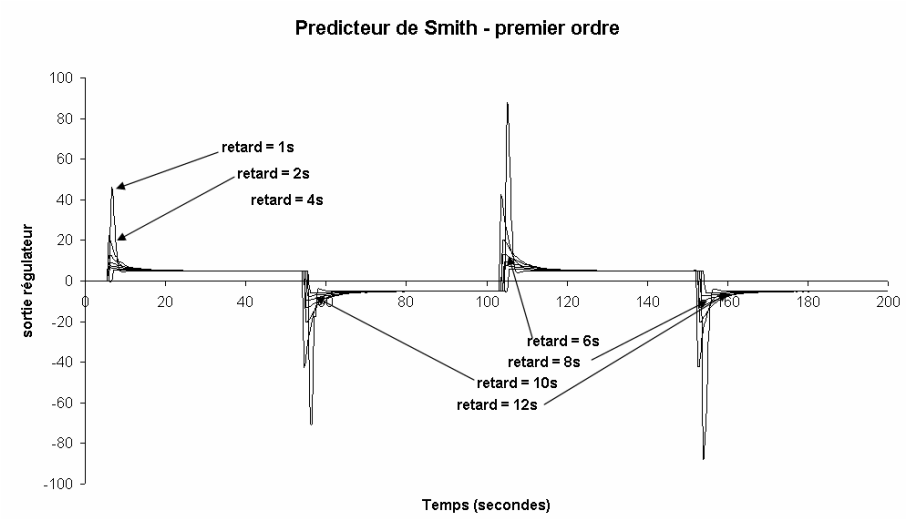


Figure 51 – Correction avec un Prédicteur de Smith d'un système stable du premier ordre avec retard – relevé 1 – sortie du régulateur (commande)

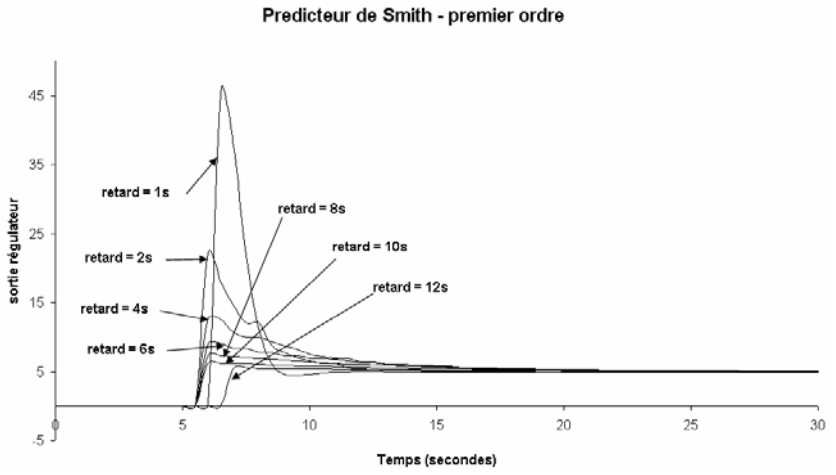


Figure 52 – Correction avec un Prédicteur de Smith d'un système stable du premier ordre avec retard – relevé 2 – sortie du régulateur (commande)

Les relevés des essais successifs pour des retards différents sont clairs : le retard est considéré d'autant plus que celui-ci est long. Les commandes prennent en compte le délai de réaction du système par la mise en place d'une commande dite 'anticipatrice'. Cette fonction se retrouve dans la structure même du régulateur avec le compensateur de temps mort (CTM).

Correction d'un système stable du deuxième ordre avec retard

Soit un système du premier ordre de la forme :

$$G(p) = \frac{2.0e^{-\tau p}}{(1+2p)^2}$$

[6.1. 02]

Les relevés des figures sont effectués pour des retards croissants (de 1 seconde à 10 secondes) et avec un temps d'échantillonnage de 500ms :

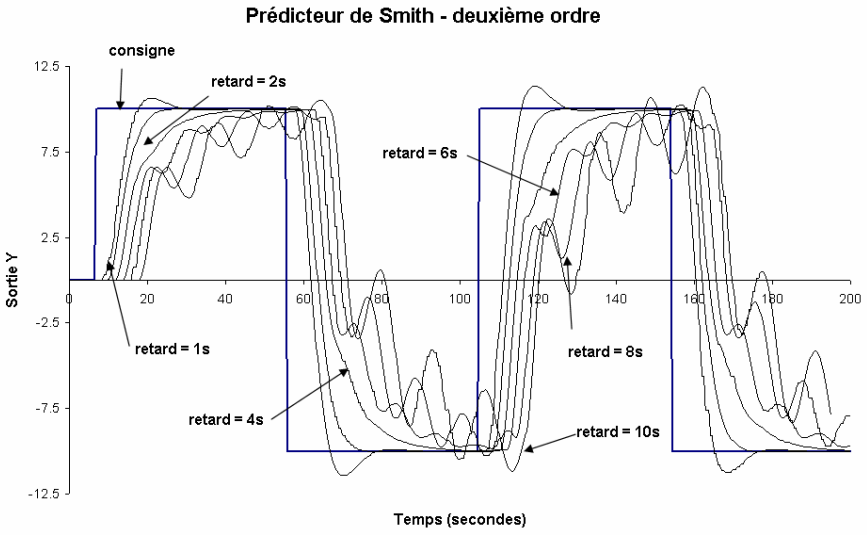


Figure 53 – Correction avec un Prédicteur de Smith d'un système stable du deuxième ordre avec retard – relevé 1 – sortie du système (mesure)

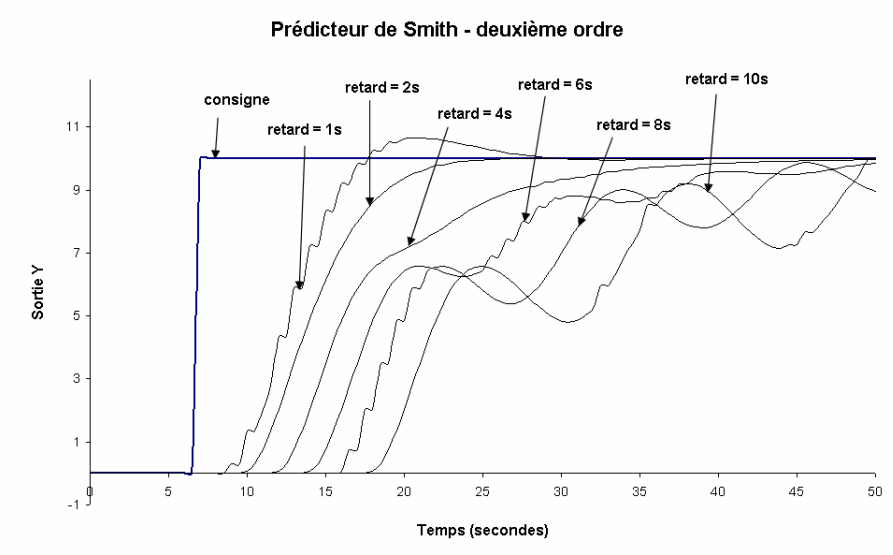


Figure 54 – Correction avec un Prédicteur de Smith d'un système stable du deuxième ordre avec retard – relevé 2 – sortie du système (mesure)

Tout comme pour le cas d'un système du premier ordre, le Prédicteur de Smith pour second ordre permet de diminuer (ou même d'annuler) les effets indésirables du retard. Il apparaît cependant que cette correction soit sensiblement moins efficace par rapport aux résultats désirés. On remarque très clairement que le système corrigé converge mais avec une stabilisation plus lente et avec quelques dépassements.

Les commandes appliquées au système sont illustrées dans les courbes suivantes :

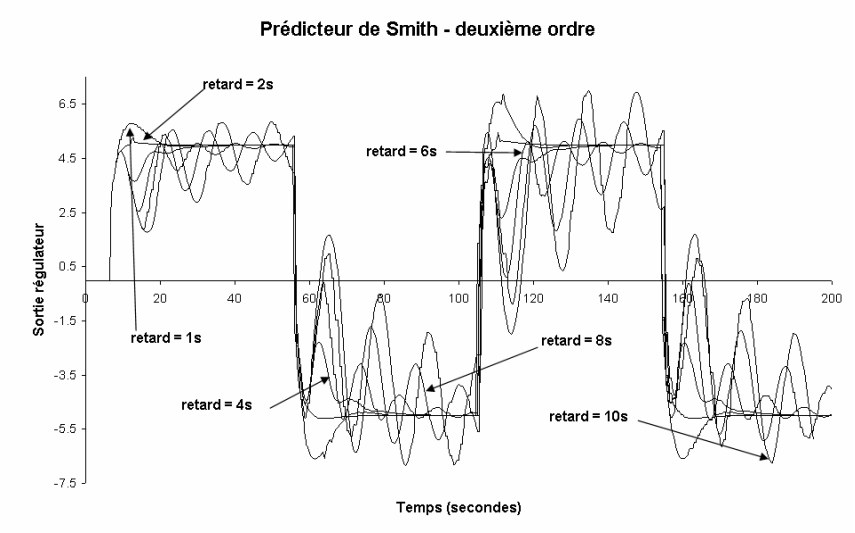


Figure 55 – Correction avec un Prédicteur de Smith d'un système stable du deuxième ordre avec retard – relevé 2 – sortie du régulateur (commande)

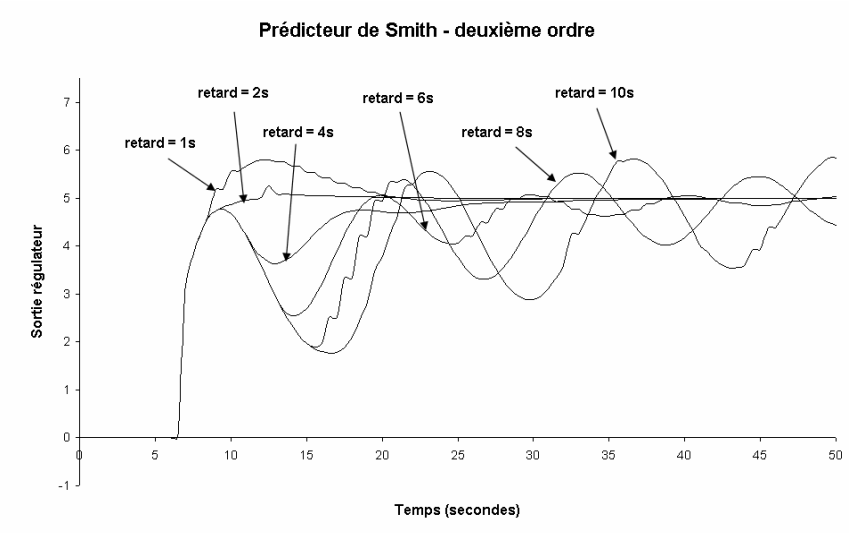


Figure 56 – Correction avec un Prédicteur de Smith d'un système stable du deuxième ordre avec retard – relevé 1 – sortie du régulateur (commande)

La fonction d'anticipation est présente mais est moins performante. Le CTM assure une action pour limiter les effets du retard mais apparemment ne remplit pas son rôle à 100%.

Cet exemple est intéressant car il met en avant les difficultés rencontrées avec l'application pratique du Prédicteur de Smith. Ceci s'explique par la nature même de l'automate et du caractère discret de la correction. L'automate programmable industriel n'est pas strictement fait pour assurer un échantillonnage parfait. Il faut toujours prendre en compte la taille de l'application. En effet, l'exécution du cycle rajoute une incertitude supplémentaire sur l'échantillonnage. Enfin le caractère discret introduit indéniablement une 'instabilité' relative lorsque l'on est dans des systèmes rapides nécessitant un fort échantillonnage.

Correction d'un système intégrateur (instable) avec retard

Soit un système intégrateur ordre de la forme :

$$G(p) = \frac{0.2e^{-tp}}{p} \qquad [6.1.\ 03]$$

Les relevés des figures sont effectués pour des retards croissants (de 1 seconde à 10 secondes) et avec un temps d'échantillonnage de 500 ms et une constante de temps en boucle fermée de 5 secondes:

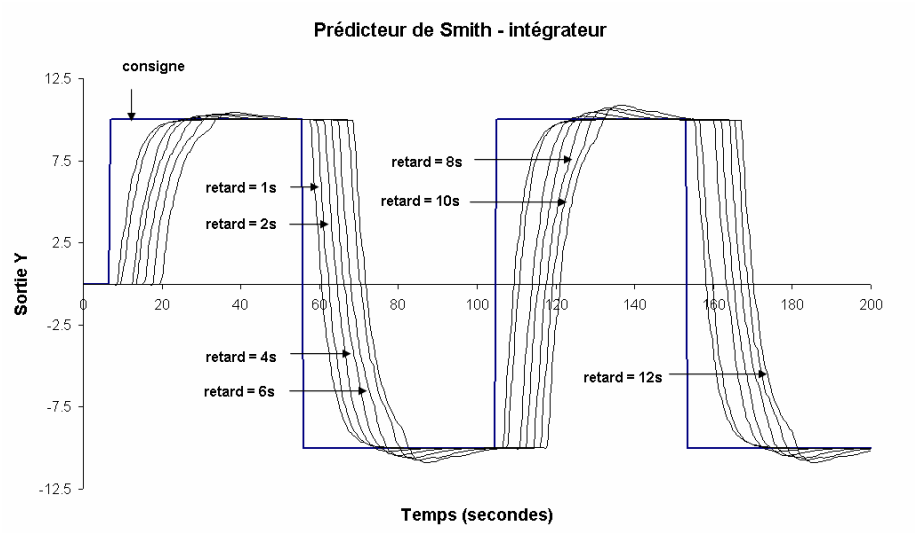


Figure 57 – Correction avec un Prédicteur de Smith d'un système intégrateur avec retard – relevé 1 – sortie du système (mesure)

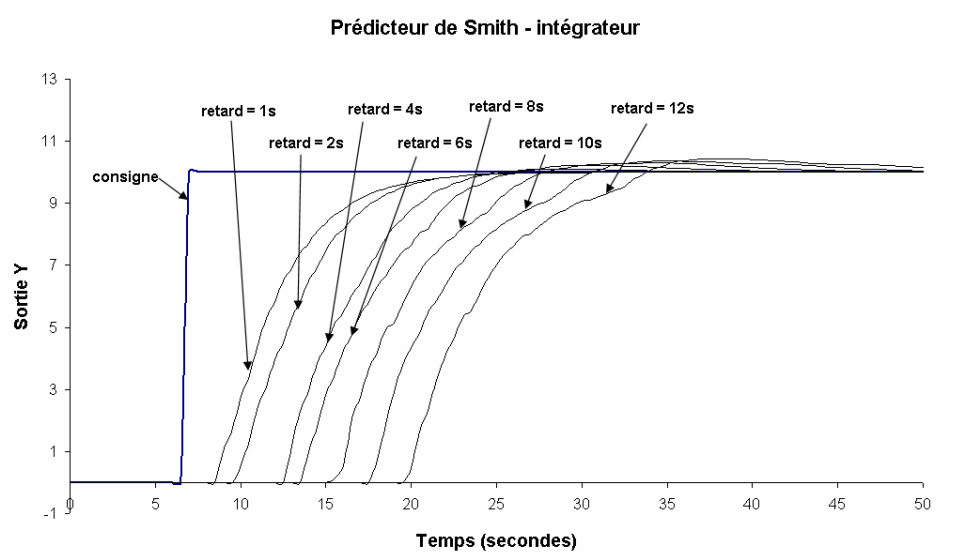


Figure 58 – Correction avec un Prédicteur de Smith d'un système intégrateur avec retard – relevé 2 – sortie du système (mesure)

Cet exemple de correction avec le Prédicteur de Smith modifié [Mat96] montre la puissance de la méthode. En dépit de légers dépassements dus principalement aux problèmes d'exécutions discrètes (évoquées précédemment), les réponses des systèmes soumis à des écarts de consignes sont concluantes. Les systèmes bouclés sont stables et précis.

Les commandes correspondantes permettent de mieux apprécier ces résultats :

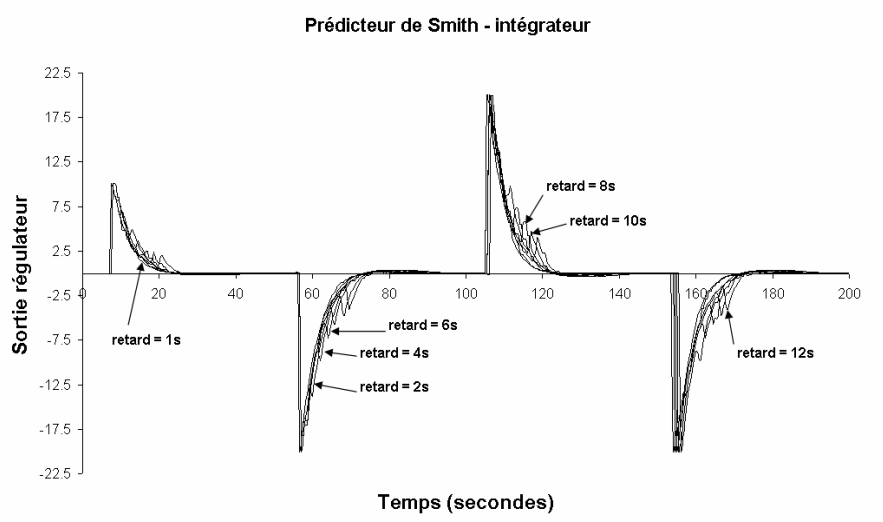


Figure 59 – Correction avec un Prédicteur de Smith d'un système intégrateur avec retard – relevé 1 – sortie du régulateur (commande)

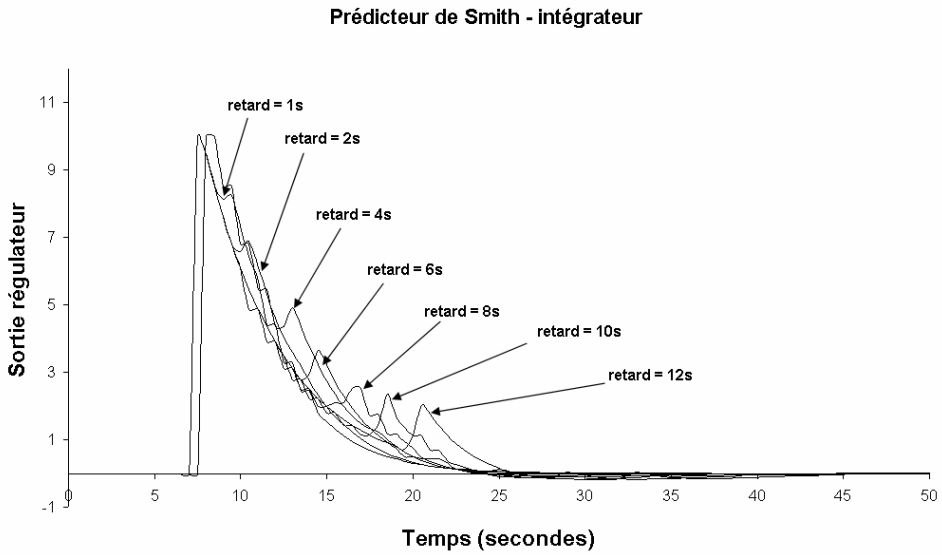


Figure 60 – Correction avec un Prédicteur de Smith d'un système intégrateur avec retard – relevé 2 – sortie du régulateur (commande)

Le Prédicteur de Smith pour système instable produit une correction adaptée. Les résultats des dernières courbes montrent que les commandes appliquées anticipent les effets désagréables du retard : les commandes ont des allures de dérivées qui traduisent ce comportement. Enfin le réglage du correcteur permet de spécifier la dynamique de sortie du système en boucle fermée. On remarque en effet que le système met environ 15 secondes pour atteindre la consigne (trois fois la constante de temps spécifiée).

Correction d'un système stable du premier ordre lent fortement retardé

Cet exemple permet de montrer l'efficacité du Prédicteur de Smith dans des cas extrêmes de très forts retards. Soit un système du premier ordre décrit par :

$$G(p) = \frac{2.0e^{-1200p}}{1+3600p} \qquad [6.1.\ 04]$$

Soit le relevé de la sortie du système et du régulateur (échantillonnage de 1 seconde):

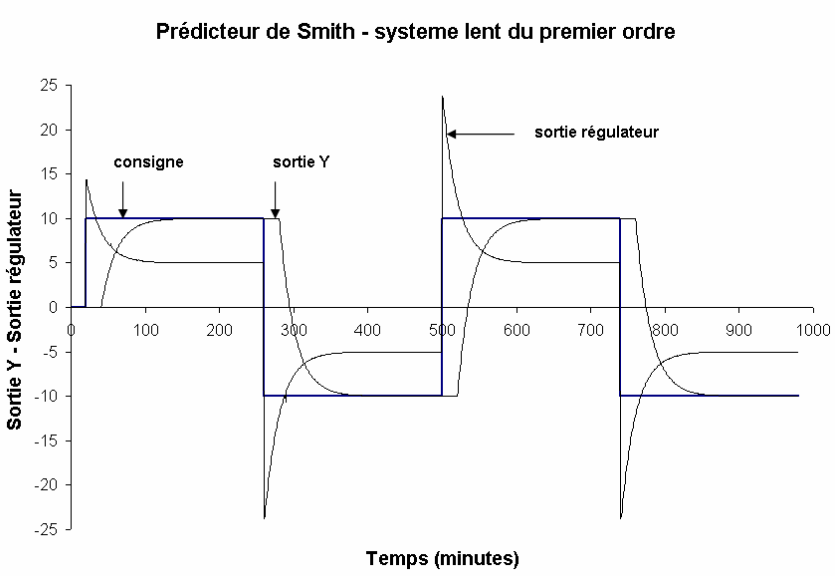


Figure 61 – Correction avec un Prédicteur de Smith d'un système stable du premier ordre lent fortement retardé - sortie régulateur et sortie système.

La sortie du système en boucle fermée est précise et converge vers la consigne désirée sans dépassements apparents. Malgré un long retard de prés de 20 minutes, le régulateur est capable de produire une commande qui s'adapte pour 'effacer' les effets du retard.

On note également que l'échantillonnage étant très important au regard de la nature du système, on ne distingue aucun des effets néfastes constatés dans les exemples précédents (dépassements légers, commandes sensiblement oscillantes, etc…).

Le Prédicteur de Smith possède une structuration simple pour la compensation du retard. Sa mise en œuvre pour des systèmes stables (premiers et seconds ordres) et systèmes instables, est tout à fait adaptée pour un automate programmable industriel.

6.2. La commande prédictive fonctionnelle généralisée

La PFC généralisée possède cet avantage de pouvoir construire un correcteur à partir de la réponse indicielle du système à corriger. La condition nécessaire et que le système doit être auto stable. A partir d'un relevé basique, il est possible par un calcul simple (exécuté dans l'objet automate) de mettre en œuvre un régulateur grâce à la représentation pour convolution. Ensuite, il suffit de définir un horizon et une trajectoire de référence cohérents pour obtenir un bon comportement en boucle fermée.

Correction d'un troisième ordre

Soit un système du troisième ordre décrit par:

$$G(p) = \frac{2.0}{24p^3 + 26p^2 + 9p + 1} \qquad [6.2.\ 01]$$

Pour la mise en œuvre de la PFC généralisée, il est essentiel de modéliser le système avec une représentation MA. L'algorithme du régulateur (PFCGene_sc) permet de s'ffranchir des calculs usuels de la commande : il suffit de donner en paramètre d'entrée, les y_i échantillonnés mesurés soumis à un échelon appliqué en entrée du système. Grâce au relevé (voir figure), on trouve un échantillonnage de 2.5 secondes:

y_1=0	y_6=1.57	y_{11}=1.97
y_2=0.11	y_7=1.74	y_{12}=1.99
y_3=0.49	y_8=1.85	y_{13}=1.99
y_4=0.93	y_9=1.91	y_{14}=2.0
y_5=1.30	y_{10}=1.95	y_{15}=2.0

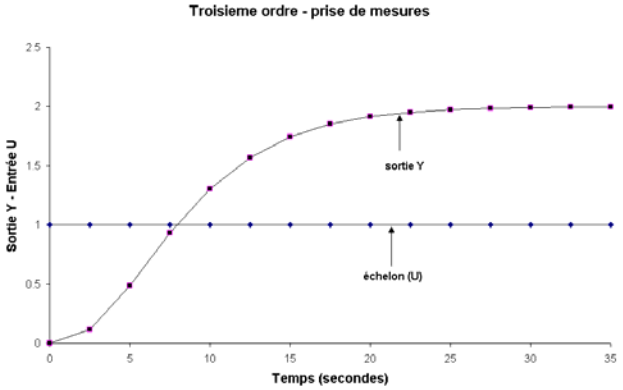


Figure 62 – PFC généralisée – troisième ordre - réponse indicielle du système pour la détermination des y_i

Avec un échantillonnage de 2.5 secondes, un horizon de 5 secondes et une constante de temps de 20 secondes pour la trajectoire de référence, on obtient les courbes suivantes :

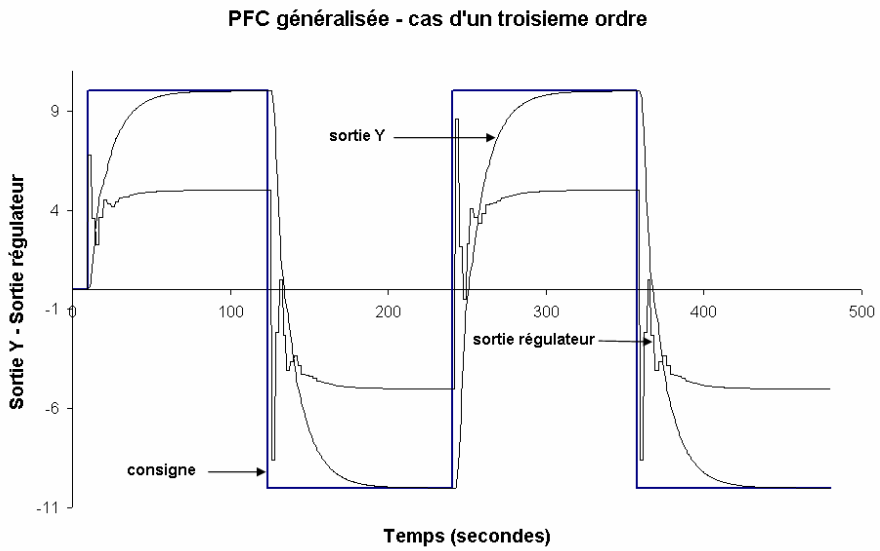


Figure 63 – Correction avec la PFC généralisée – troisième ordre – sortie régulateur et sortie système.

La sortie du régulateur est calculée suivant les performances désirées. On remarque bien que le système atteint la consigne en environ 60 secondes. Le système bouclé est précis et n'a pas de dépassement.

Correction d'un quatrième ordre – système lent

Soit un système du quatrième ordre décrit par :

$$G(p) = \frac{2.0}{15000000p^4 + 6250000p^3 + 87500p^2 + 500p + 1} \qquad [6.2.\ 02]$$

Le relevé des y_i échantillonnés mesurés suite à une réponse indicielle (échantillon de 75 secondes) est :

y_1=0	y_8=1.21	y_{15}=1.92
y_2=0.02	y_9=1.40	y_{16}=1.94
y_3=0.10	y_{10}=1.56	y_{17}=1.96
y_4=0.27	y_{11}=1.68	y_{18}=1.97
y_5=0.50	y_{12}=1.77	y_{19}=1.98
Y_6=0.74	y_{13}=1.84	Y_{20}=1.99
Y_7=0.99	y_{14}=1.89	

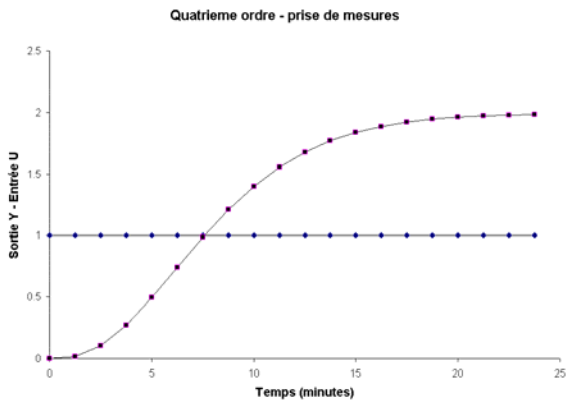


Figure 64 – PFC généralisée – quatrième ordre - réponse indicielle du système pour la détermination des y_i

Avec un échantillonnage de 75 secondes, un horizon de 150 secondes et une constante de temps de 10 minutes pour la trajectoire de référence, on obtient les courbes suivantes :

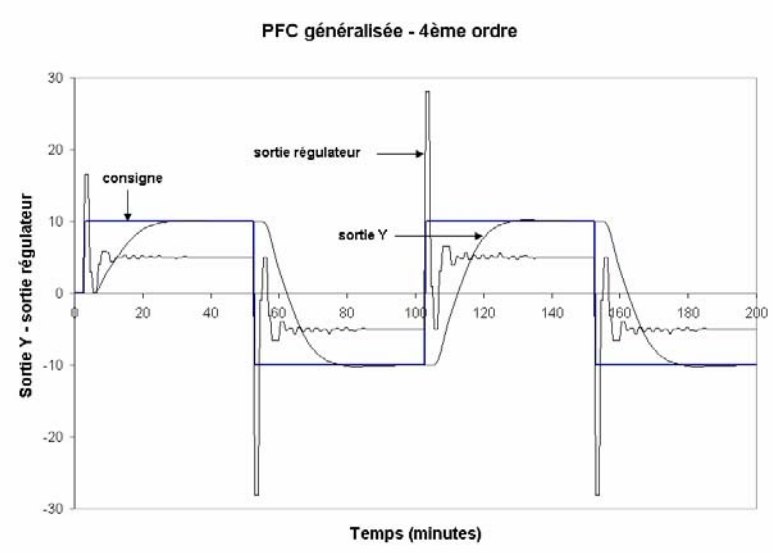


Figure 65 – Correction avec la PFC généralisée – quatrième ordre – sortie régulateur et sortie système.

La sortie du système bouclée est stable et 'raccroche' la consigne en environ 25 minutes. Il y a peu de dépassement et la sortie du régulateur converge sans soucis vers sa valeur de stabilisation.

La PFC généralisée et un exemple concret de la puissance d'une méthode prédictive pour une large palette de systèmes auto stables, et nécessitant un moindre coup de mise en œuvre. L'automate programmable industriel est tout à fait adapté pour cette application.

6.3. La commande prédictive généralisée

Les résultats ici présentés portent sur la correction des mêmes systèmes de la section sur la PFC généralisée [6.2]. En pratique, la GPC est avantageuse dans la mesure où l'on peut construire un correcteur sous forme RST.

Correction d'un troisième ordre

Soit le système [5.2. 01]. La représentation discrète de ce modèle est également :

$$G(z) = \left. \frac{0.1126z^{-1} + 0.2334z^{-2} + 0.02904z^{-3}}{1 - 1.256z^{-1} + 0.5105z^{-2} - 0.06665z^{-3}} \right|_{T_e = 2.5s}$$

[6.3. 01]

L'application des algorithmes de commande, pour la mise en place de la GPC pour trois horizons différents, est résumée dans la table ci dessous:

Cas 1	**Cas 2**	**Cas 3**
N_1=1	N_1=1	N_1=1
N_2=3	N_2=5	N_2=8
N_u=1	N_u=1	N_u=1
λ=1.11	λ=5.27	λ=15.37
T_e système = 10 ms	T_e système = 10 ms	T_e système = 10 ms
T_e régulateur = 2.5 s	T_e régulateur = 2.5 s	T_e régulateur = 2.5 s
Polynôme RST= R=[2.55 -2.77 1.04 -0.13] S=[1.00 -0.94 0.43 -0.48] T=[0.05 0.22 0.42] $T_{conseil}$=0.69	Polynôme RST= R=[1.83 -2.15 0.84 -0.11] S=[1.00 -0.95 0.36 -0.41] T=[0.01 0.05 0.09 0.12 0.14] $T_{conseil}$=0.41	Polynôme RST= R=[1.56 -1.89 0.75 -0.097] S=[1.00 -0.96 0.33 -0.38] T=[0.00 0.02 0.03 0.04 0.05 0.06 0.06 0.06] $T_{conseil}$=0.32

Remarque : (Pour la détermination du polynôme T, voir [Cam])

Les relevés qui suivent mettent en évidence les comportements de la sortie du système et du régulateur :

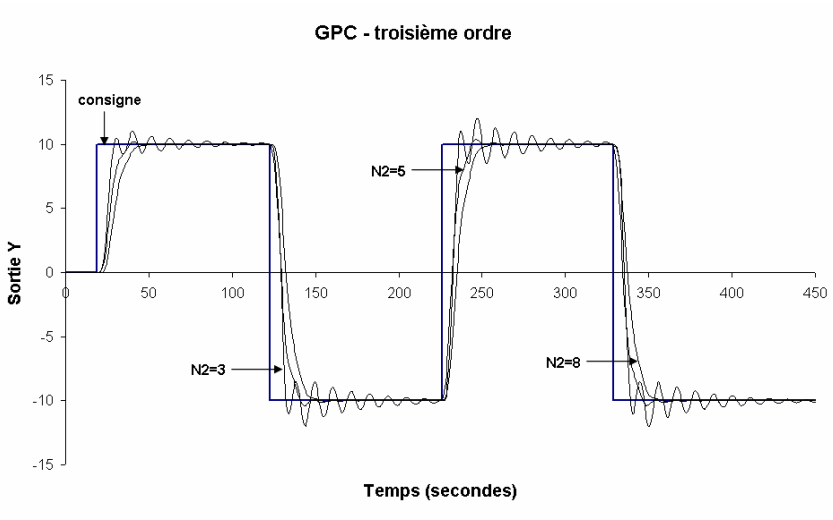


Figure 66 – Correction avec la GPC – troisième ordre – sortie système.

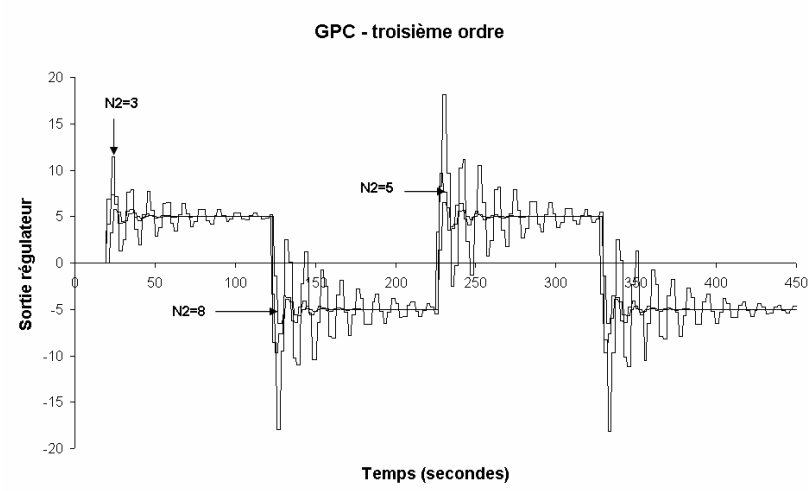


Figure 67 – Correction avec la GPC – troisième ordre – sortie régulateur.

La sortie du système bouclée est dépendante des paramètres de réglage de la GPC. Les courbes précédentes montrent l'impact de l'horizon fuyant de prédiction pour le système corrigé : la sortie du régulateur converge plus rapidement vers un état d'équilibre et la sortie bouclée est plus stable lorsque N_2 augmente. Une grande valeur de ce paramètre ne garantit pas pour autant de bonnes performances mais ces courbes mettent en avant le caractère prédictif de la GPC au travers de la définition de l'horizon du Prédicteur optimal. Enfin, ces résultats donnent un premier aperçu concluant sur la mise en œuvre pratique de la GPC dans l'automate programmable industriel.

Correction d'un quatrième ordre

Soit le système [5.2. 02]. La représentation discrète de ce modèle est également :

$$G(z) = \frac{0.01627z^{-1} + 0.05466z^{-2} + 0.01162z^{-3} + 1.805e-006z^{-4}}{1 - 2.044z^{-1} + 1.428z^{-2} - 0.3423z^{-3} + 2.681e-014z^{-4}}\Bigg|_{T_e=75s}$$

[6.3. 02]

Les paramètres pour deux horizons différents ayant permis la mise en œuvre de la GPC sont synthétisés dans la table ci-dessous :

Cas 1	**Cas 2**
N_1=1	N_1=1
N_2=8	N_2=20
N_u=1	N_u=1
λ=5.27	λ=47.00
T_e système = 30 s	T_e système = 30 s
T_e régulateur = 75 s	T_e régulateur = 75 s
Polynôme RST= R=[8.29 -16.05 10.73 -2.47] S=[1.00 -1.00 0.08 0.38 -0.46] T=[], $T_{conseil}$=0.5	Polynôme RST= R=[6.47 -13.07 9.05 -2.15] S=[1.00 -1.00 0.07 0.34 -0.41] T=[], $T_{conseil}$=0.29

Les relevés suivants mettent en évidence les comportements de la sortie du système et du régulateur :

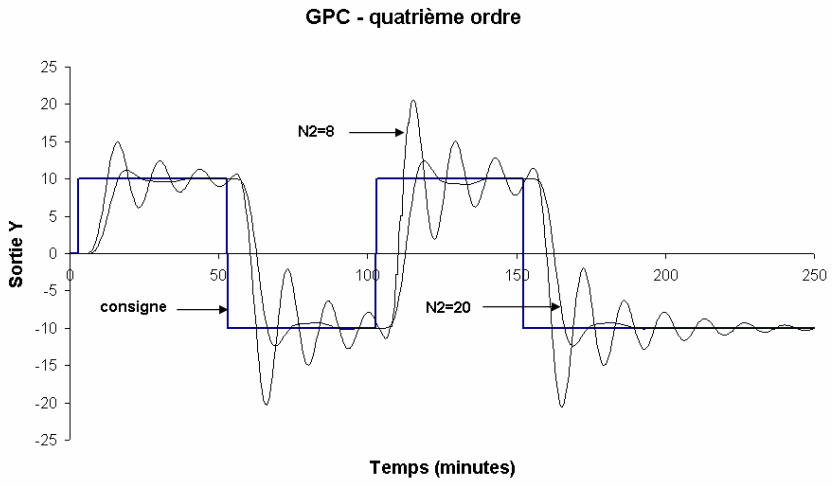


Figure 68 – Correction avec la GPC – quatrième ordre – sortie système.

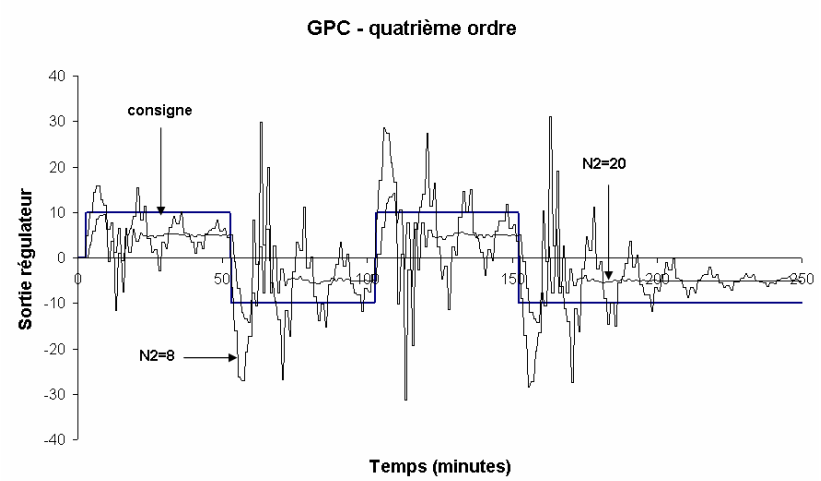


Figure 69 – Correction avec la GPC – quatrième ordre – sortie régulateur.

Tout comme pour l'exemple précédent, la sotie du système bouclé est dépendante de l'horizon du prédicteur optimal. Les commandes obtenues convergent plus rapidement pour un large horizon.

La GPC est une commande prédictive qui est capable de réaliser le contrôle de nombreux systèmes par la mise en action d'un prédicteur optimal soumis à un critère quadratique de minimisation. Sa puissance réside dans une mise en œuvre non restrictive de tout processus modélisé par un processus ARMA. Sa limitation se retrouve dans des calculs matriciels susceptibles d'être lourds pour un automate et par la résolution d'équations diophantiennes pouvant être complexes.

6.4. Le correcteur RST

Le correcteur RST a une structure permettant l'implémentation de nombreux régulateurs différents. Il permet notamment l'élaboration des correcteurs assurant des performances désirées en boucle fermée par le placement de pôles et l'imposition d'une trajectoire de poursuite.

Placement de pôle et modèle de référence pour la poursuite

Soit le procédé décrit par le modèle discrétisé H (Exemple pris de [Lan] et adapté):

$$H_{sys}(z) = \left. \frac{-0.009627z^{-1} + 0.01637z^{-2}}{1 - 1.895z^{-1} + 0.8972z^{-2}} \right|_{T_{sys}=0.5s}$$

[6.4. 01]

Soit le modèle H_m de référence pour la dynamique de poursuite :

$$H_m(z) = \left.\frac{0.0927z^{-1} + 0.0687z^{-2}}{1 - 1.2451z^{-1} + 0.4066z^{-2}}\right|_{T_{sys}=4s}$$

[6.4 02]

Et le polynôme P de l'identité de Bézout décrivant la dynamique de régulation (pôles : 0.6225 +/- j0.138) :

$$P(z) = 1 - 1.3741z^{-1} + 0.4867z^{-2}$$

[6.4. 03]

Le contrôleur RST, correspondant à ces spécifications, est calculé pour un cycle de 4 secondes et est tel que :

$$R(z) = 3 - 3.94z^{-1} + 1.31z^{-2}$$
$$S(z) = 1 - 0.37z^{-1} - 0.63z^{-2}$$
$$T(z) = 3.33 - 4.59z^{-1} + 1.62z^{-2}$$

[6.4. 04]

On obtient les relevés des sorties du correcteur RST et du système H_{sys} en boucle fermée :

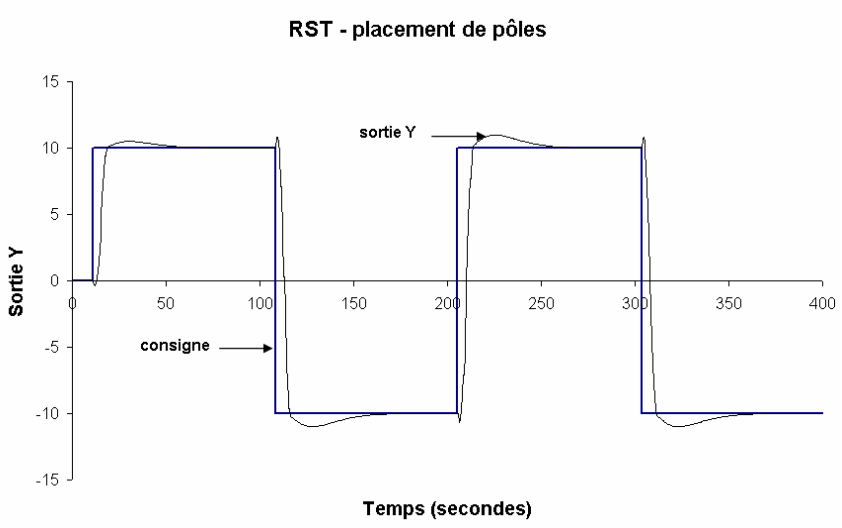


Figure 70 – Correction RST – Placement de pole avec modèle de poursuite – sortie système

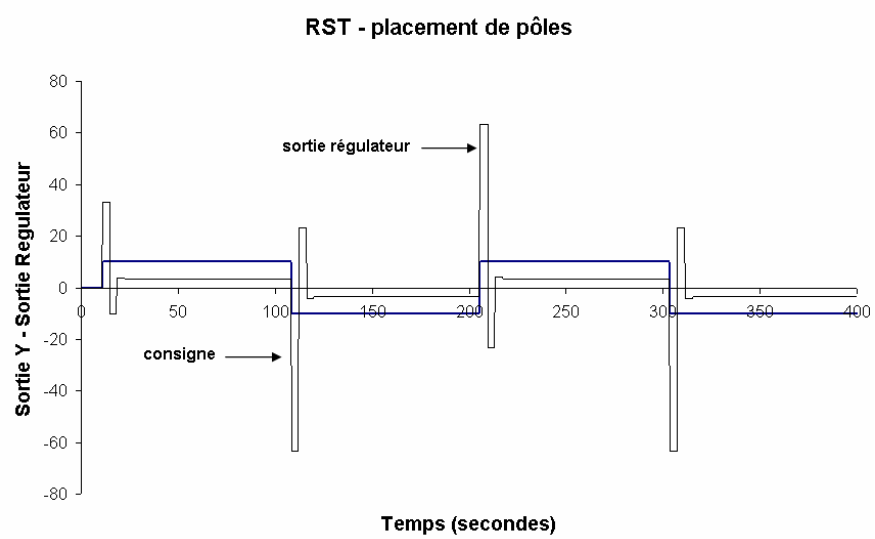


Figure 71 – Correction RST – Placement de pôles avec modèle de poursuite – sortie régulateur

Les courbes précédentes semblent toutes indiquer que le placement de pôles sous automate Schneider est performant et fonctionne normalement.

Dans cet exemple il est intéressant de faire la simulation sous Matlab. Ainsi il est possible de comparer le comportement de la régulation sous automate avec les résultats souhaités. La courbe suivante faite ce comparatif sur la sortie :

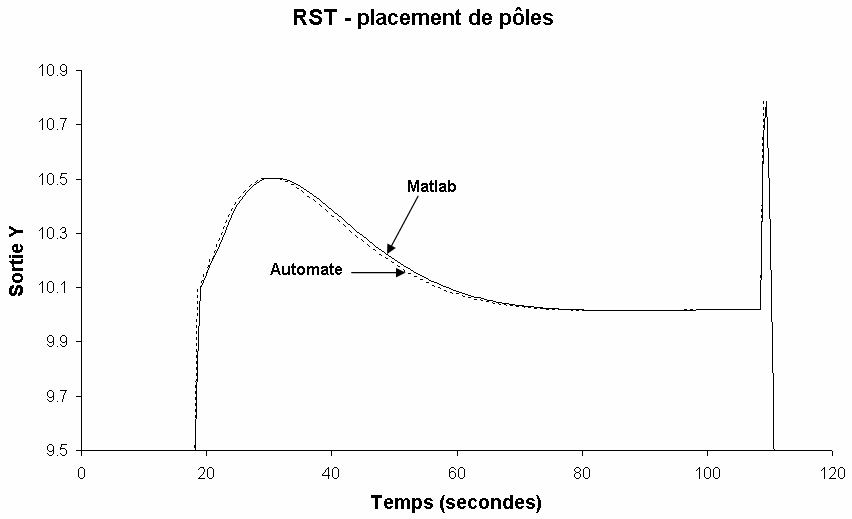


Figure 72 – Correction RST – Placement de pôle avec modèle de poursuite – Comparaison de la sortie système sous Matlab et sous Automate.

On constate que les résultats entre simulation (Off Line) et Automate (On line) sont quasi identiques. Malgré certains désagréments liés au fonctionnement même de l'automate (synchronisme) cet exemple de placement de pôle accrédite l'implémentation.

La synthèse d'un correcteur RST, dans le cas d'un placement de pôles, reste une entreprise complexe d'automaticien. Cependant, elle offre la possibilité d'imposer certains comportements avec des performances désirées.

L'automate programmable se prête bien à cette structure polynomiale. Le correcteur RST possède cet avantage de pouvoir représenter nombre de régulateurs sous forme polynomiale, d'où un intérêt supplémentaire pour l'automate.

6.5. Récapitulatif

Le tableau qui suit fait un résumé synthétique des caractéristiques des régulations utilisées pour les systèmes à gaz :

	Avantages	Inconvénients	Utilisation
PID	- simple à mettre en place - présent dans la plupart des automates - résout beaucoup de situations de régulations	- fausse simplicité d'utilisation - limité pour des problèmes plus complexes	- sur systèmes ayant peu de retard et pouvant être modélisés par un premier ordre ou un intégrateur.
Prédicteur de Smith	- simple à implémenter dans l'automate. - simple à utiliser - permet de fortement atténuer (voir éliminer) les effets indésirables du retard - robuste	- n'est pas applicable à tout système même avec retard. - quelques effets gênants pour des systèmes rapides (problème d'échantillonnage)	- bonne solution pour des systèmes moyennement rapides ayant des retards significatifs (premier ordre, deuxième ordre pôle double et systèmes intégrateurs).
PFC	- commande prédictive basée sur des principes simples à comprendre. - bibliothèque de base disponible sous Schneider (bibliothèque PCR) : premier ordre, deuxième ordre, troisième ordre, système intégrateur. - une solution généralisée pour tous système auto stable d'un ordre quelconque. - robuste et permet dans certains cas d'éliminer	- implémentation automate peu instinctive	- bonne solution de remplacement du PID lorsque des problèmes de dépassements et de performances sont rencontrés

	complètement les dépassements		
GPC	- commande prédictive qui est généralisable à tout système ARMA. - solution pouvant être modélisée avec la représentation polynomiale du correcteur RST	- algorithme très complexe à implémenter dans l'automate (limitation dans l'ordre du modèle à résoudre) - réglage nécessitant un niveau d'expertise fort.	- solution dans les cas où la PFC ne peut résoudre le problème
RST	- simple à implémenter dans l'automate - structure ouverte pouvant être utilisée pour la modélisation d'autres régulateurs - une solution pouvant fixer le placement de pôles et le suivi d'une trajectoire de référence	- niveau expert d'utilisation	- solution pratique pour la mise en œuvre de la commande GPC - solution faite pour la spécification de performance par le placement de pôles et la poursuite (robustesse et dynamique de la réponse)

Figure 73 - Tableau récapitulatif des régulations implémentées dans l'automate programmable pour les systèmes à gaz.

7. Conclusion

La correction des systèmes à gaz peut être faite par plusieurs stratégies de contrôle-commandes. L'intégration d'algorithmes avancés sous forme d'objets dans l'automate programmable Schneider, est un moyen efficace de systématiser leurs utilisations pour le projet GCS. Le Prédicteur de Smith peut résoudre des problématiques dues aux retards. La commande prédictive fonctionnelle (PFC) est adaptée à l'implémenter dans les systèmes numériques. Le contrôleur RST possède une structure ouverte qui est directement transposable dans l'automate, permettant une utilisation qui relève de l'expertise en automatique. Enfin la commande prédictive généralisée peut être employé pour tout système modélisé avec une représentation ARMAX.

Dans ce contexte les algorithmes que j'ai introduis dans l'automate Schneider répondent à une palette de cas susceptibles d'être rencontrés pour le projet Gaz. Les résultats obtenus dans la dernière partie de ce chapitre illustre la validité de l'implémentation.

CHAPITRE 4

INTEGRATION DES ALGORITHMES DE CONTROLES AVANCES AVEC DEUX METHODOLOGIES NOUVELLES DE L'INFORMATIQUE INDUSTRIELLE

Chapitre 4. Intégration des algorithmes de contrôles avancés avec deux méthodologies nouvelles de l'informatique industrielle

Ce chapitre explique les stratégies évoluées de réalisation du contrôle-commande pour le projet GCS. J'y expose les choix du projet et j'y aborde l'intégration du contrôle avancé dans ce contexte innovant. Pour ce faire j'utilise une approche objet. Au travers d'une démarche complexe mais efficace, j'aborde le design mis en place pour introduire un objet nommé MultiController. Celui-ci répond à des problématiques de réalisation du contrôle commande d'un niveau supérieur. Le MultiController que j'ai développé concrétise l'objectif de la thèse.

1. Les méthodologies nouvelles pour l'informatique industrielle

L'informatique industrielle est un terme générique qui désigne l'ensemble des technologies informatiques qui touchent de prés ou de loin le génie industriel associé aux automatismes. Wikipedia, l'encyclopédie libre du web apporte également une définition intéressante à ce sujet: "L'informatique industrielle est une branche technologique de l'informatique appliquée qui couvre l'ensemble des techniques de conception, d'analyse et de programmation de systèmes à base d'interfaçage de l'informatique avec de l'électronique, électrotechnique, mécanique, robotique etc à vocation industrielle[...]".

Le génie informatique est ce que l'on pourrait appeler la "théorie pure" pour l'automaticien. Le 'process control' et la production d'applications pour automates programmables ne se préoccupent guère des concepts de haut niveau déjà très répandus dans la conception logicielle. Cette vision de deux mondes séparés est cependant en train de se modifier.

Deux orientations majeures se distinguent :

- L'utilisation d'un cadre de développement permettant la réalisation plus efficace des applications process control: le framework. La réutilisation systématique d'entités (approche 'objet') étant introduite.
- L'approche de développements dirigée par les modèles.

1.1. Le framework

Le framework est un terme provenant de l'ingénierie du développement logiciel. Il désigne un canevas de travail et s'attache aux outils et méthodologies pour la réalisation complète ou partielle d'un projet.

Un framework est un design réutilisable de tout ou d'une partie d'un système qui est représenté par des classes abstraites et par leurs utilisations interactives. C'est un squelette d'application qui peut être personnalisé par l'utilisateur [Joh]. On parle également de cadriciel qui « est un ensemble de classes abstraites collaborant entre elles pour faciliter la création de tout ou d'une partie d'un système logiciel. Un cadriciel fournit un guide architectural en partitionnant le domaine visé en classes abstraites et en définissant les responsabilités de chacune ainsi que les collaborations entre classes. […] » [Wik].

La déclinaison des besoins en méthodes, en outils et en techniques pour le développement (dans des processus de créations softwares) provient d'une offre des produits graduellement plus abondante sur le marché [Mat00]. Ainsi le framework ne se limite plus uniquement aux métiers du développement informatique mais s'exporte bien au delà.

1.2. Le Model Driven Engineering et le Model Driven Architecture

Le Model Driven Engineering (MDE) s'attache à l'utilisation de modèles pour la description et le design. Plus couramment dédié au monde du développement logiciel, le MDE considère les modèles comme entités de premier niveau.

Fin 2001, le groupe OMG (Object Management Group) a développé le concept MDA (Model Driven Architecture) qui est une application directe du MDE. La philosophie du MDA est une solution à base de modèles pour s'affranchir des difficultés classiques des plateformes de développements.

Le MDA est une nouvelle perspective de représentation et de développent par l'utilisation d'un modèle logique pour le traduire en un modèle physique [OMG][OMG03][OMG01][OMG00]. Ainsi une application est une représentation d'un modèle physique conforme à un méta modèle au-dessus de lui [LET].

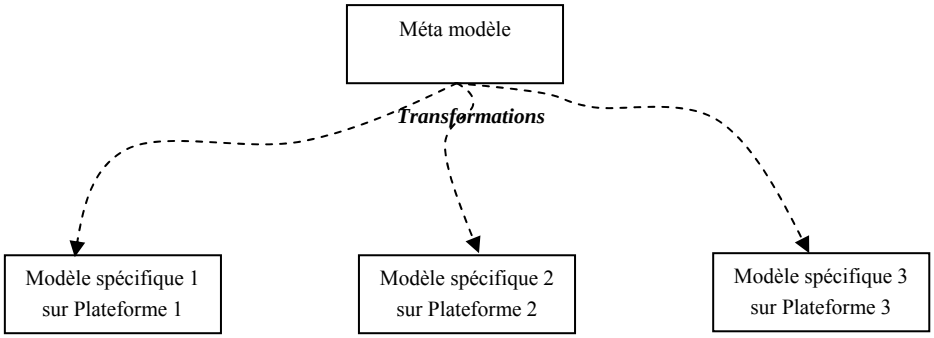


Figure 74 – MDA : Méta modèle et modèles spécifiques

Le MDA définit aussi le principe de dépendance vis à vis de la plateforme du modèle : PIM (Platform Independant Model) et PSM (Platform Specific Model).

1.3. Enseignements pour le contrôle commande

Les systèmes de contrôle industriel utilisent pleinement les technologies PLC associées à un SCADA. Ce choix d'architecture de contrôle d'équipement a fait ses preuves et de nombreuses possibilités gravitant autour de cette solution ont vu le jour. Ainsi, le chef de projet a à sa disposition de multitudes décisions pour déterminer le choix des fournisseurs (pour le PLC, le SCADA, les bus de terrains ...) et pour construire son architecture de contrôle (distribuée, locale, décentralisée ...) suivant les éléments de ressources obtenus. Les outils de développement d'automatisation évoluent rapidement. L'offre s'étoffe et met à disposition des plateformes diversifiées pour assurer les fonctions de contrôle commande. Ainsi les développements sont soumis à plus de complexité et à plus de méthodologies.

De ce fait le framework tel qu'il est défini peut être assimilé par le contrôle commande comme une standardisation des outils, des procédures et des utilisations nécessaires au développement des applications.

L'approche MDD ou MDA pour le contrôle commande est une solution adéquate aux difficultés rencontrées lors de migrations d'une application d'une plateforme à une autre. L'utilisation de modèles de description est une solution adaptée pour s'affranchir des problèmes classiques de changement de technologies : la migration sur d'autres plateformes de développement en devient plus efficace.

1.4. Une nouvelle gestion de projet

La qualité d'un projet d'automatisation provient de sa capacité à répondre aux besoins du client en lui fournissant une installation robuste clés en main.

L'intérêt de l'utilisation d'un framework de développement et de l'approche dirigée par les modèles réside principalement en la réflexion nécessaire en amont pour la réalisation finale. Cela ne signifie pas que le développement sans ces deux méthodologies est voué à un mauvais résultat. Ces deux solutions (prises séparément également) peuvent être assimilées à des facilitateurs permettant de répondre à une bonne partie des besoins du projet.

En dépit d'une liberté de choix relative, la réalisation d'un système de contrôle industriel n'est plus uniquement une réalisation ponctuelle dans l'entreprise. On ne veut plus refaire le déjà fait. On ne veut plus gaspiller les compétences humaines. On cherche à réutiliser les expériences passées, à construire sur des bases qui ont déjà fait leurs preuves. En résumé, le système de contrôle doit être performant et réutiliser les valeurs ajoutées développées dans le groupe de travail et le reste de l'entreprise. Le framework et le MDD apportent des solutions concrètes car elles répondent à une recherche de standardisation orientée vers le développement. Ce sont des solutions alternatives à une succession de choix stratégiques pour capitaliser les ressources de développement par la réutilisation et la standardisation. Ces deux orientations font partie d'une démarche stratégique qui s'intègre pleinement dans la gestion de projet de contrôle commande.

1.5. Implication : l'approche objet dans le PLC

Les algorithmes avancés de régulation sous automates programmables industriels sont implémentés avec la norme IEC61131-3 relative aux langages de programmation.

L'utilisation efficace et pertinente de cette norme est la 'fabrication' d'objets assurant une utilisation standard, garantie par un seul code source, et pour toutes les instances susceptibles d'être implémentées pour un même projet.

Cette approche dite 'objet' pour le PLC est assurée par la construction astucieuse d'une entité appelée objet DFB (Diagram Functional Bloc). Le DFB permet de créer un objet unique non spécifique dans l'un ou plusieurs des langages de la norme. Il est ensuite utilisé dans la séquence d'exécution globale sous un nombre non limité d'instances spécifiques. Cette caractéristique essentielle de la programmation automate se révèle très judicieuse dans un projet où framework et MDD sont susceptibles d'être utilisés. Elle

permet de hiérarchiser et de découper le problème dans le code automate. L'objet PLC peut être mis en relation à la notion de composant, (qui doit établir des relations et des connexions pour ses besoins et ses responsabilités (notion de *ports*)) [LET].

La régulation avancée encapsulée dans un objet DFB est ainsi implémentée sous une forme unique. Elle est donc réutilisable sans restriction de nombre (si ce n'est les limitations en mémoire de l'automate). Cet avantage de traiter le problème par une structuration basée sur les composants est indéniable dans un processus de production à grande échelle. Le développement à base de modèles en est simplifié et le framework est aussi très bien adapté à cette configuration.

2. Le framework UNICOS et le Model Driven du projet gaz au CERN

Le projet GCS (Gas Control System) du LHC est un projet initialisé en 2000. Il a pour but de fournir 23 systèmes de contrôle pour les systèmes à gaz des détecteurs et sous détecteurs des expériences du LHC [LHC].

Une installation à gaz permet de délivrer une mixture constante (à des conditions de fonctionnement déterminés) dans une enceinte précise. Les systèmes à gaz sont réalisés pour durer sur du long terme.

Le LEP (ancien accélérateur avant le LHC) des années 90 avait 25 systèmes à gaz pour l'ensemble des expériences. L'ensemble de la réalisation de ces 25 systèmes de contrôle était réparti en quatre équipes indépendantes. Plus de quatre systèmes de contrôle à gaz complètement différents furent développés, et mis en œuvre. Ces solutions diverses et variées étaient donc incompatibles de par leur conceptions et leurs technologies.

Les motivations furent donc d'avoir une approche commune pour le LHC et ses installations de contrôle à gaz. Le CERN recommanda d'utiliser des technologies industrielles (Bus de terrain, Automates et SCADA) via le groupe de travail JCOP [JCO].

2.1. Les objectifs et solutions du projet gaz

Le premier objectif du projet fut de fournir une Interface Homme Machine (HMI : Human Machine Interface) pour l'ensemble des installations de manière à homogénéiser le fonctionnement et la maintenance. Il fut également demandé d'intégrer les applications dans DCS (Detector Control System) des expériences du LHC. Le second objectif fut de réduire les efforts de travail (développement, maintenance et opération) aux vues des similitudes des 23 systèmes à gaz.

Pour répondre aux objectifs du projet, des concepts, des procédures d'opérations, des standards internes communs, etc. furent choisis. Rapidement, il fut convenu de travailler avec une approche framework basée sur des solutions techniques industrielles afin de réduire les efforts de développement et de maintenance.

2.1.1. Le raisonnement par options

La première initiative fut de définir une approche globale pour l'ensemble des 23 systèmes. Chaque installation fut définie comme une combinaison d'un système de référence qui décline toutes les possibilités d'une installation à gaz. Par une mise en place d'un raisonnement optionnel, on peut identifier les équipements et éléments constitutifs de l'installation.

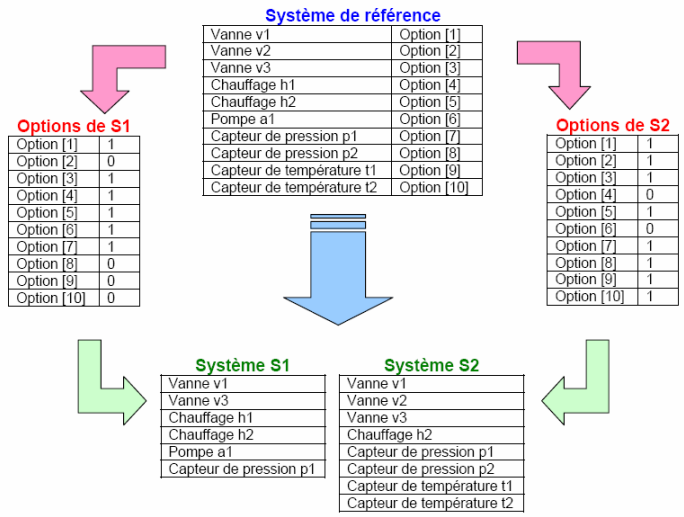


Figure 75 – Raisonnement par options : exemple de principe

Ce raisonnement en cascade s'inscrit complètement dans l'approche framework. Les équipements sont codifiés par des options afin d'être facilement identifiables. Ils sont standards au projet, ce qui apporte un gain non négligeable pour la maintenabilité et l'opération sur site.

2.1.2. Une approche modulaire du système à gaz

La grande force du projet GCS réside en une solution modulaire de l'installation. Chaque système est une combinaison de modules possédant un certain nombre d'options. Une installation se décline par la présence ou non de modules tels que : le Mixer, la Distribution, l'Exhaust, le Purifier, Le CO_2 removal, la Pump, l'Analyse, etc. Chaque module est constitué d'options d'équipements qui sont ou non actives.

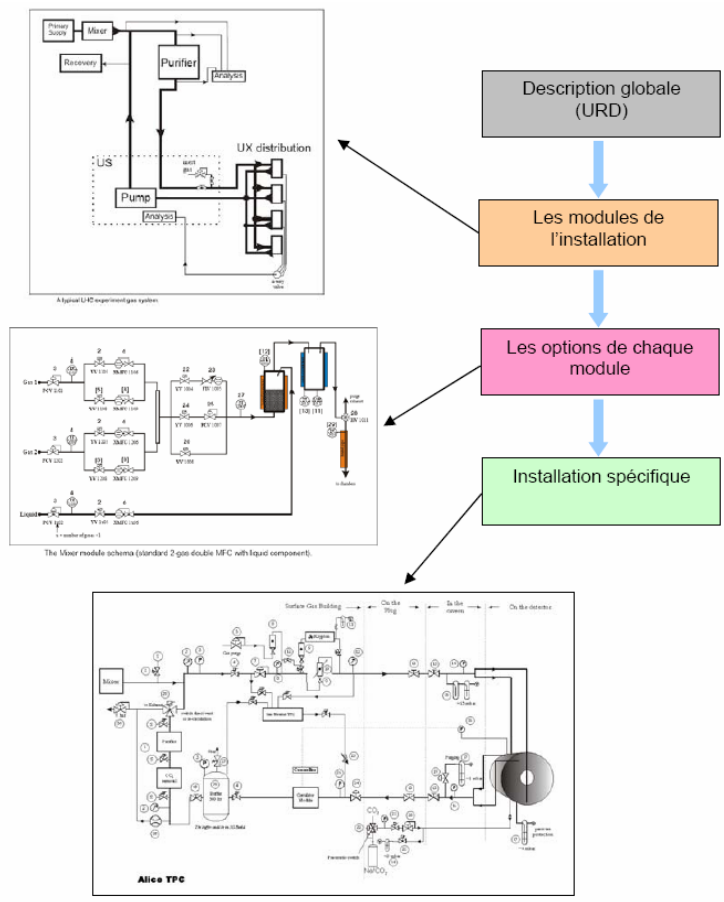


Figure 76 – L'approche modulaire du système à gaz

2.1.3. Une analyse fonctionnelle s'intégrant dans le framework de développement

Un document de référence URD (User Requirement Document) fut produit afin de permettre le bon travail de réalisation des systèmes de contrôle. Ce document est une analyse fonctionnelle relative au 23 installations. Il prend en compte l'ensemble des options et des possibilités puis synthétise les besoins des experts gaz. Cet URD est non seulement une transcription des besoins du client mais surtout une formulation claire et précise des fonctionnalités que doivent réaliser l'ensemble des installations de contrôle des systèmes à gaz.

2.1.4. Des solutions industrielles

Le SCADA choisi est PVSS d'ETM. Ce système de supervision est fourni par le framework JCOP du CERN. Ce choix rentre dans une démarche de globalisation de l'opération du LHC.

Les PLC sont de marque Schneider. Plus particulièrement, le projet GCS utilise les Premiums associés à Unity. La programmation choisie est le langage ST (Structured Text) assurant une bonne visibilité dans la démarche et avec les outils du projet mis en place.

2.1.5. Utilisation d'un framework existant : UNICOS

Le projet utilise également UNICOS (UNified Industrial COntrol System) qui est un framework de travail développé par le CERN et supporté par JCOP dans PVSS [UNI]. UNICOS est composé principalement par :

- Une bibliothèque de programmation PLC.
- Un middleware dédié pour les systèmes de contrôle industriel du CERN utilisant une association PLC/SCADA (automates de Schneider et PVSS d'ETM) avec son 'baseline' (projet de base vierge).
- Des widgets et faceplates standards de supervision (sous PVSS).
- Un générateur d'instances (pour Concept et Unity).

Le framework UNICOS est construit avec une hiérarchisation à trois niveaux pour le développement d'application de contrôle dans le PLC.

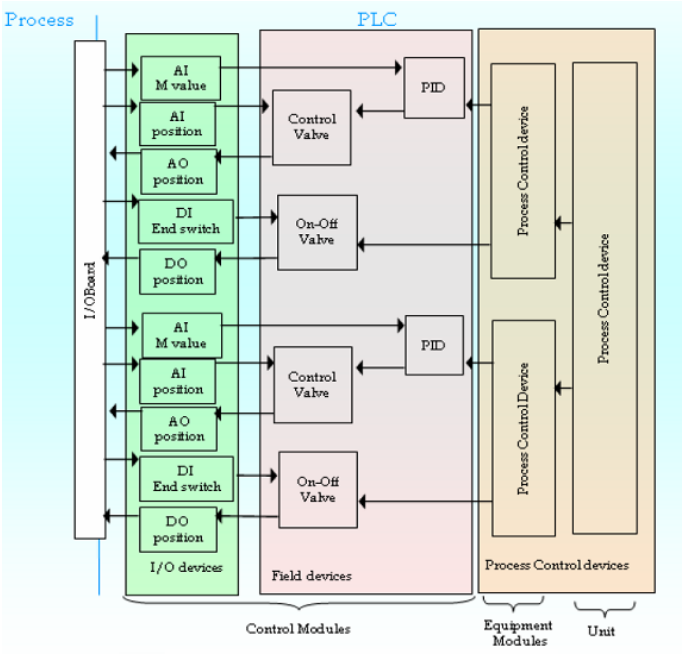


Figure 77 – La hiérarchisation à trois niveaux dans le PLC avec le framework UNICOS

Remarque : Dans les années 90, Jacobson a introduit une théorie à propos de l'approche objet dans le contrôle [Jac]. Il explique qu'en définissant des objets sur trois niveaux, les fonctions et les dépendances sont mieux exprimées et assurent une meilleure maîtrise du développement. Les fonctionnalités qui sont directement dépendantes du système sont mises dans l'objet d'interface. Le stockage et la manipulation des informations sont encapsulés dans l'objet entité et les informations spécifiques qui ne peuvent pas être placées naturellement ailleurs, sont introduites dans l'objet de contrôle. UNICOS est de très prés un exemple de la théorie développée par Jacobson.

2.1.6. Des outils informatiques d'aide à la gestion du projet

Le groupe de travail utilise quelques outils informatiques performants d'aide à la gestion du projet GCS :

- WinCVS est utilisé pour l'organisation et le maintien des informations informatiques (gestion des versions).
- Savannah est également utilisé pour l'archivage et la résolution des bugs trouvés dans le projet.
- Remedy a été choisi pour le suivi des requêtes utilisateurs.

Ces choix permettent une meilleure qualité et un suivi plus précis du projet. Ils font partie de la stratégie globale du projet.

2.2. Le framework et l'approche Model Driven du projet Gaz

2.2.1. Le LHC GCS framework

La caractéristique principale du framework gaz est la production automatique d'une application de l'interface homme machine (HMI) jusqu'au code automate [GCS03]. Il utilise le framework UNICOS mais bénéfice d'autres outils et méthodologies supplémentaires, ce qui lui confère sa spécificité propre. Il peut être divisé en deux parties : l'une réservées pour le développement PLC et l'autre pour le développement SCADA.

2.2.1.1. La génération automatique du code automate

On peut distinguer deux aspects dans la génération de code PLC. Tout d'abord, on travaille avec la déclaration et l'instanciation des composants (DFB). Cette partie de code est du niveau instance et reflète le design de l'installation (instanciations des objets Process d'une librairie par exemple). Il y a ensuite la partie logique du Process qui est

l'image des exigences fonctionnelles du processus vues par le client. C'est la partie la plus sujette à des modifications.

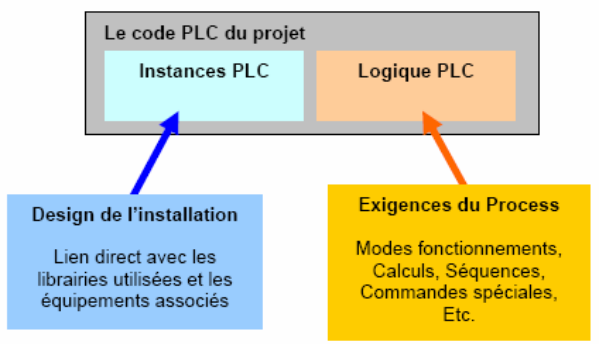


Figure 78 – Le code automate dans un automate pour le projet gaz

L'approche framework dans la génération de code PLC est liée aux bibliothèques d'utilisation du projet GCS. L'application de la norme IEC6131-3 introduit une notion de répétitivité. Dans cette vision orientée objet, le programmeur dispose d'une bibliothèque type qui lui permettra d'instancier ses équipements spécifiques de la même manière : par exemple un objet <TypeVanne> pour chaque vanne du système. Ils sont dépendants des équipements et des composants du framework LHC GCS. La logique, quant à elle se limite à la manipulation des variables associées aux objets automates instanciés. Elle assure aussi toutes les fonctions et les conditions spécifiques de l'automatisme dans son ensemble.

Le framework fournit deux outils majeurs pour la génération PLC : le générateur d'instances (*Instance Generator*) pour les instances codées du projet, ainsi que le générateur de logique (*Logic Generator*) pour le code process. Durant la phase de production, ces outils utilisent des fichiers d'entrées spécifiques et les traitent en créant des fichiers d'imports directement chargeables dans les plateformes de développement automate (UnityV2.3 pour le premium de Schneider). Le code automate est remis en forme pour assurer une communication cohérente (middleware) avec le SCADA (PVSS), ainsi l'application PLC est prête.

2.2.1.2. La génération automatique du HMI

La génération automatique du SCADA utilise des outils développés via le framework LHC GCS. Ils permettent de générer tous les panneaux, les synoptiques, les faceplates, les widgets, etc. nécessaires pour l'application finale [GCSf05].

Des composants SCADA remplacent la configuration interactive usuelle du niveau supervision d'une application automatisée.

L'application spécifique est réalisée avec six composants (data-driven components) construits sur le principe d'une génération faite grâce à un outil appelé *XML Generator* :

- Le composant synoptique (synoptic view component)
- Le composant courbe de tendance (trend view component)
- Le composant navigation fenêtre (window tree component)
- Le composant recettes (recipe component)
- Le composant anomalie (anomaly component)
- Le composant rapport des alertes (alert summary component)

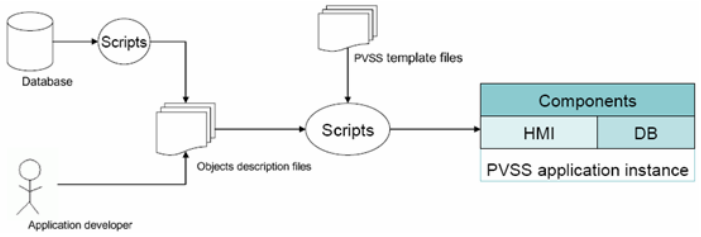


Figure 79 – Principe de générations des composant SCADA du framework LHC GCS

2.2.1.3. Résumé

La figure ci-dessous résume simplement le principe du framework LHC GCS avec les outils associés dans le processus de génération automatique de l'application finale spécifique :

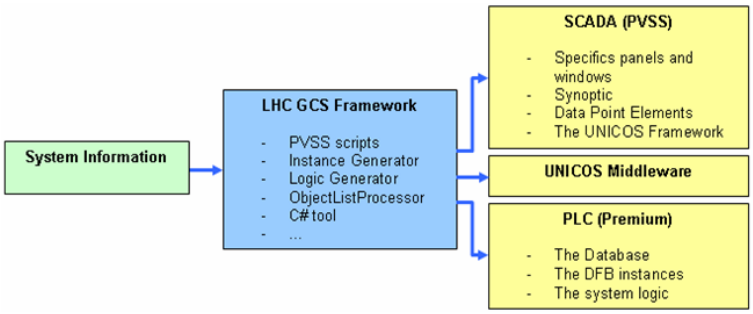


Figure 80 – Principe simple du framework LHC GCS et de ses outils

2.2.2. Le développement dirigé par les modèles (Model Driven) du projet gaz

Le développement dirigé par les modèles est une étape supplémentaire du projet GCS dans sa mise en œuvre d'automatiser l'ensemble de la production d'une application [GCSmd05]. Cette stratégie globale se greffe au-dessus du framework GCS mais est toujours en lien étroit avec celui-ci.

L'idée principale est l'addition d'outils supplémentaires au framework pour obtenir une approche contenant un modèle de référence (méta modèle) pour construire tous les modèles spécifiques (23 pour le projet GCS). Ces derniers sont directement interprétables pour la production automatique 'classique' dans le framework.

Proche du MDA, cette solution est pourvue d'un méta modèle possédant l'ensemble des données du projet et la description précise de chacun des 23 systèmes à gaz (en termes de contrôle de processus, d'HMI etc..). Il contient aussi la description hardware ainsi que les variations (options et modules) des installations. Ce méta modèle est implémenté avec Microsoft Access. Tout ceci est réalisé par un outil appelé *Meta Data Generator*.

Enfin, un outil appelé *Object List Processor* produit les modèles spécifiques pour la production automatique des applications dans le framework.

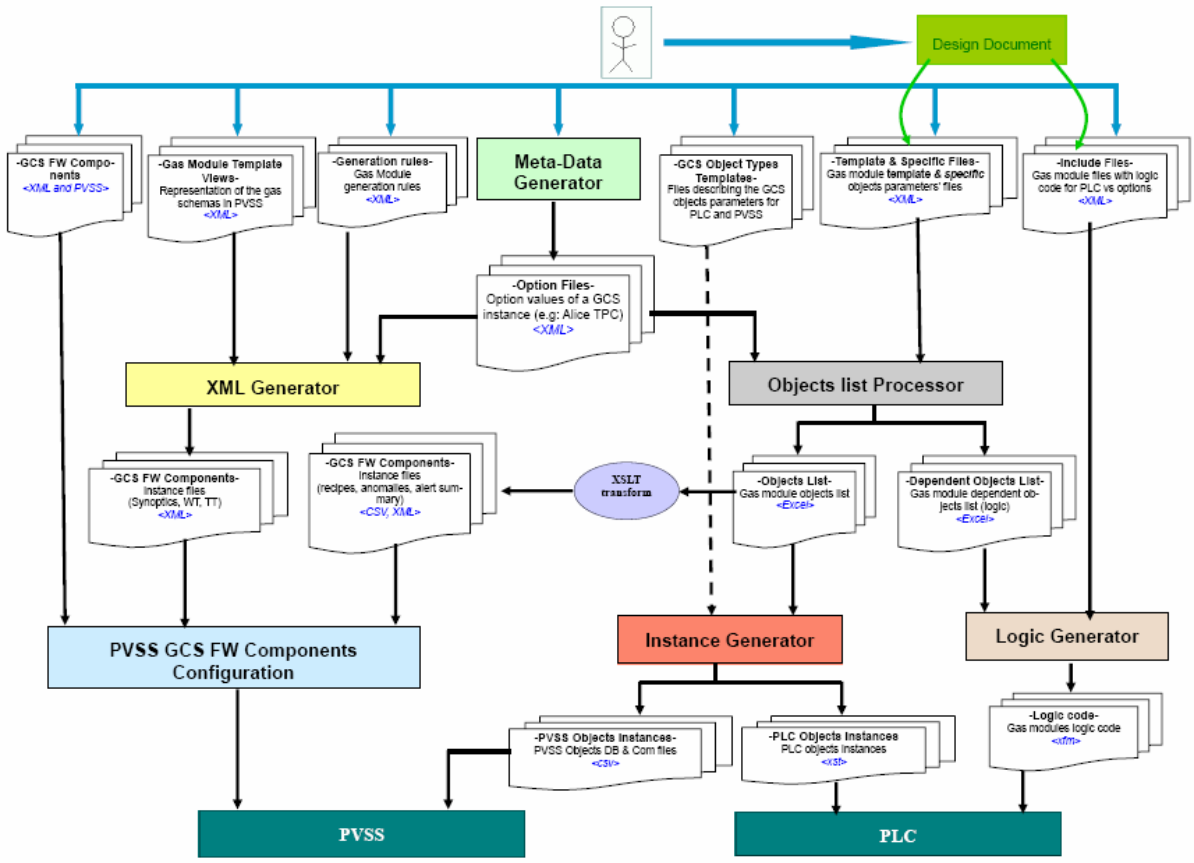


Figure 81 – framework et Model Driven pour le projet GCS

3. L'objet MultiController dans le projet gaz

Le MultiController est un objet que j'ai développé dans le projet gaz afin d'introduire des algorithmes de régulations avancées pour les installations GCS. Le MultiController est un objet UNICOS supplémentaire que j'ai produit pour le framework UNICOS. Je l'ai également intégré au framework GCS. Il est en cours d'utilisation dans le processus de génération automatique d'applications par l'approche Model Driven. C'est une application concrète d'introduction de commandes avancées dans un projet d'automatisme d'envergure, ayant des méthodologies nouvelles de l'informatique industrielle. Le MultiController assure l'intégration des algorithmes par l'établissement d'un lien virtuel objet entre le SCADA, l'automate et l'installation :

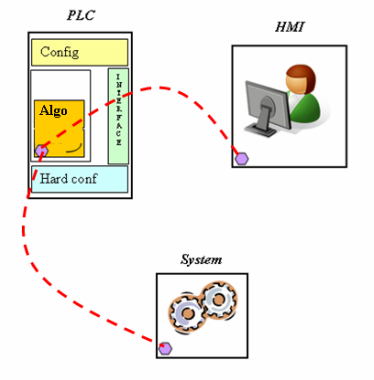


Figure 82 – Le MultiController : un moyen d'introduire des algorithmes avancés en établissant le lien entre les divers organes de l'automatisme industriel

Remarque : j'ai assuré la définition et la mise en œuvre des principes du MultiController ainsi que le développement côté automate programmable. Le développement sous PVSS du HMI a été quant à lui réalisé par Artem Burmyakov (Ingénieur de développement du projet Gaz).

3.1. Principes et fonctions du MultiController

J'ai réalisé le MultiController sur des principes de fonctionnement qui sont cohérents avec les algorithmes avancés de contrôle et qui prennent en considération les contraintes technologiques des méthodologies du projet.

En amont de l'élaboration de l'objet, j'ai mené une réflexion globale afin de satisfaire les clients. En concertation avec les acteurs concernés, j'ai traduit les exigences et j'ai établi un design de l'objet en conséquence. J'ai ensuite pu étayer les besoins des utilisateurs et converger vers une solution pertinente. L'objectif affiché était de fournir des fonctionnalités qui sont un compromis entre expertise et simplicité d'utilisation.

3.1.1. Exigences client

Etant intégré au framework UNICOS, le MultiController se soumet à des exigences spécifiques. Celles-ci sont énumérées ci-dessous :

- L'objet suit au plus près possible le standard UNICOS. Cela signifie que les formats, les fonctions et les interfaces doivent suivre au mieux ce qui est déjà défini dans les autres objets UNICOS de base.
- La sortie et la consigne peuvent être limitées.
- Un mécanisme interne peut mettre une consigne qui monte avec une rampe précisée.
- L'objet a un comportement dit de poursuite (Tracking) comme défini dans UNICOS. Lorsque le système est en régulation, la fonction Tracking peut interrompre la régulation en appliquant une commande prédéterminée (conditions et valeurs données par le client).
- Le MultiController peut faire une mise à l'échelle (Scaling) de la valeur mesurée, de la sortie du contrôleur et de la consigne.

3.1.2. Choix de design

J'ai proposé un modèle du MultiController pour satisfaire au plus juste les besoins des programmeurs, des experts et des utilisateurs. J'ai fait des choix fondamentaux de design que j'ai étudiés puis appliqués. Ils apportent des facilités (et des avantages) non négligeables à l'ensemble des personnes contribuant au développement et à l'utilisation de l'objet :

- Le MultiController utilise la notion de 'modes' pour tout ce qui a trait aux modes UNICOS. Ces modes se décomposent comme suit : le mode automatique, le mode manuel et le mode manuel forcé.
- Le MultiController utilise la notion de 'comportements' pour le fonctionnement en régulation et en position de l'objet (Positioning).
- La gestion des modes et des comportements est différente. On peut travailler en régulation ou en position dans les trois modes UNICOS.
- Le MultiController possède le PID afin de satisfaire l'ensemble des utilisateurs.
- L'objet offre plusieurs algorithmes avancés de contrôle dans une seule entité.
- Il est possible d'ajouter d'autres contrôleurs sans changer l'interface grâce à une structure DFB générique. Cette possibilité permet un gain substantiel de temps et de transparence.
- Un mécanisme de recettes existe et permet de sauver les paramètres pertinents des régulations pour être réutilisés si besoin. Cette opération s'effectue à partir de la console HMI (PVSS II).

- Les paramètres de l'objet peuvent provenir d'une implémentation avec ou sans recettes.

3.1.3. Fonctionnalités

3.1.3.1. A propos des régulations avancées dans l'objet

Au travers du MultiController j'offre la possibilité de sélectionner l'algorithme de régulation du contrôleur. Le choix peut être fait dans le programme automate ou via l'interface dédiée dans le HMI. De cette manière, l'opération est munie d'un degré de liberté appréciable en ce qui concerne la mise en place et le changement d'un algorithme de régulation avancée.

J'ai construit l'objet avec une interface DFB unique dans l'automate programmable. J'ai encapsulé dans le MultiController, toutes les régulations que j'ai développé pour l'automate Schneider (chapitre 3). J'ai choisi une structure de paramètres génériques (nommée « Param » dans l'objet) afin de réaliser une approche monolithique de l'objet. La manière dont je traite les paramètres de la structure dépend de la sélection de la régulation. Ainsi, un même paramètre peut être utilisé de différentes manières suivant l'algorithme actif du contrôleur.

Par conséquent, la structure du MultiController que j'ai définie autorise l'ajout de commandes avancées supplémentaires sans changer l'interface DFB. L'addition d'algorithmes est ainsi plus efficace dans le processus de développement et d'évolution de l'objet.

3.1.3.2. A propos des modes et des comportements du MultiController

Le MultiController travaille avec trois modes de fonctionnement (standard UNICOS) :

- Le mode automatique, dans lequel le programme a le contrôle sur l'objet.
- Le mode manuel, dans lequel l'utilisateur (HMI) prend le contrôle de l'objet ; le passage au mode automatique par requête automatique peut également s'effectuer (perte du mode manuel).
- Le mode manuel forcé, dans lequel l'utilisateur prend le contrôle total de l'objet (le mode forcé ne peut jamais être perdu sans intervention de l'utilisateur).

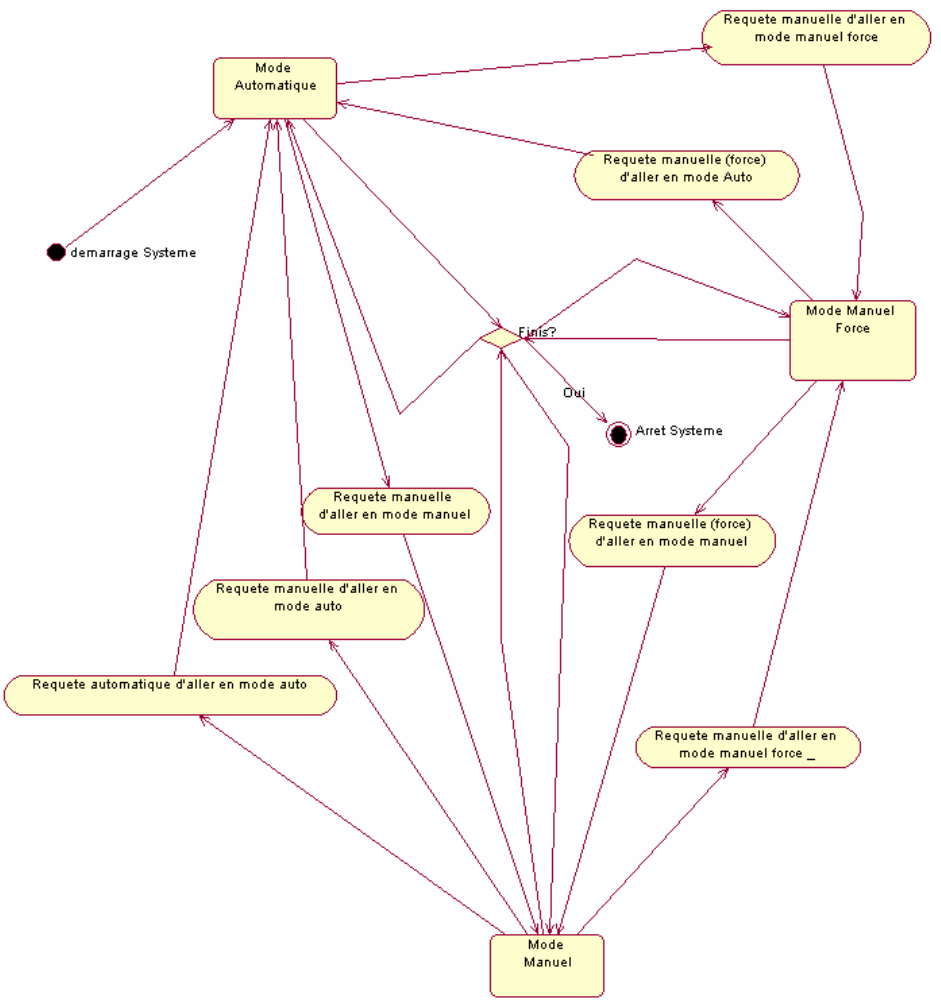


Figure 83 – Les modes du MultiController et leur fonctionnement

J'ai muni le MultiController de deux comportements différents : la régulation et le positionnement ('Regulation' et 'Positioning'). Le fonctionnement des modes avec celui des comportements est fondamentalement différent. La régulation signifie que l'objet travaille avec un type de régulation. Le positionnement est une affectation directe de la sortie du contrôleur lorsque la régulation n'est plus désirée. Cela revient à travailler en boucle ouverte et en boucle fermée.

La figure qui suit résume synthétiquement les comportements et leurs interactions avec les modes :

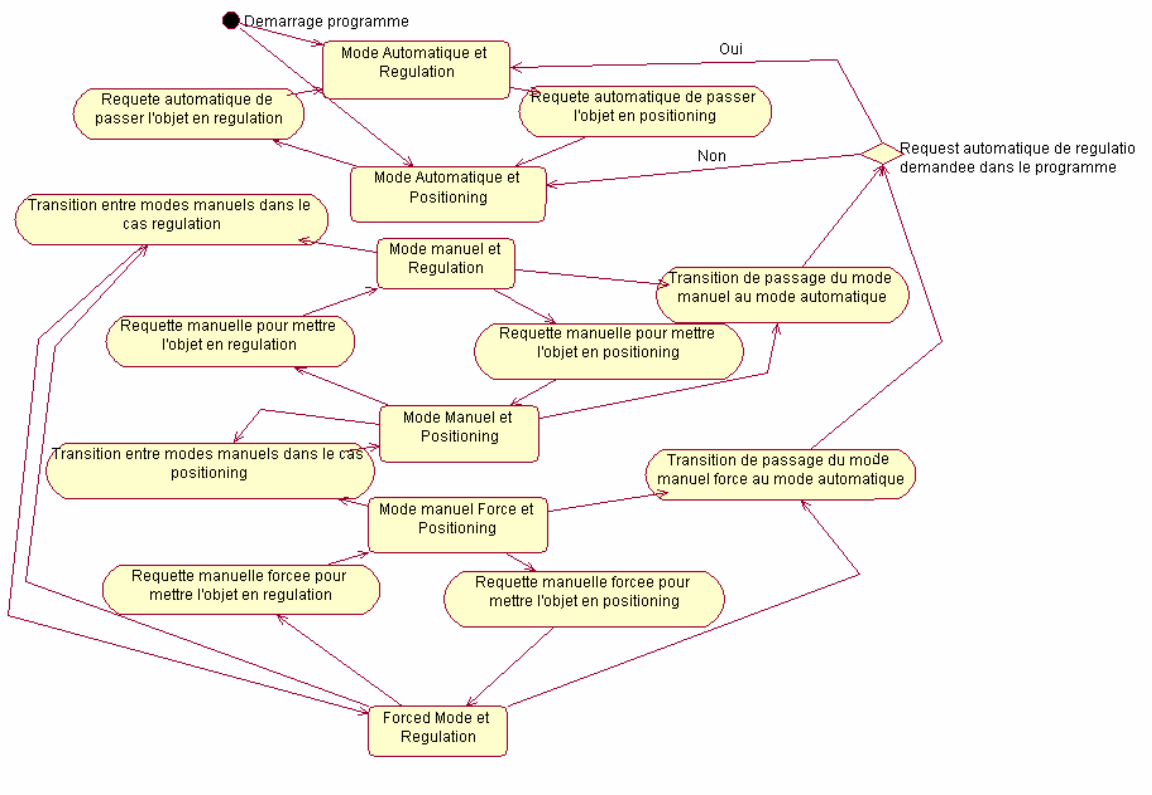


Figure 84 – Les comportements du MultiController et leur fonctionnement

Le changement de mode ne change pas le comportement, sauf dans le cas de transitions spécifiques relatives (exemple : en mode automatique, les requêtes et paramètres automatiques sont pris en priorité).

3.1.3.3. A propos du Tracking

Le Tracking est une fonctionnalité particulière que j'ai mise dans le MultiController suite au standard UNICOS. C'est une technique permettant au programme de prendre le contrôle de la sortie du contrôleur lors de la régulation, peu importe le mode actif. De cette manière, il est possible, dans des conditions bien spécifiques, de 'suivre' l'objet contrôlé et ainsi d'éviter des redémarrages violents en régulation.

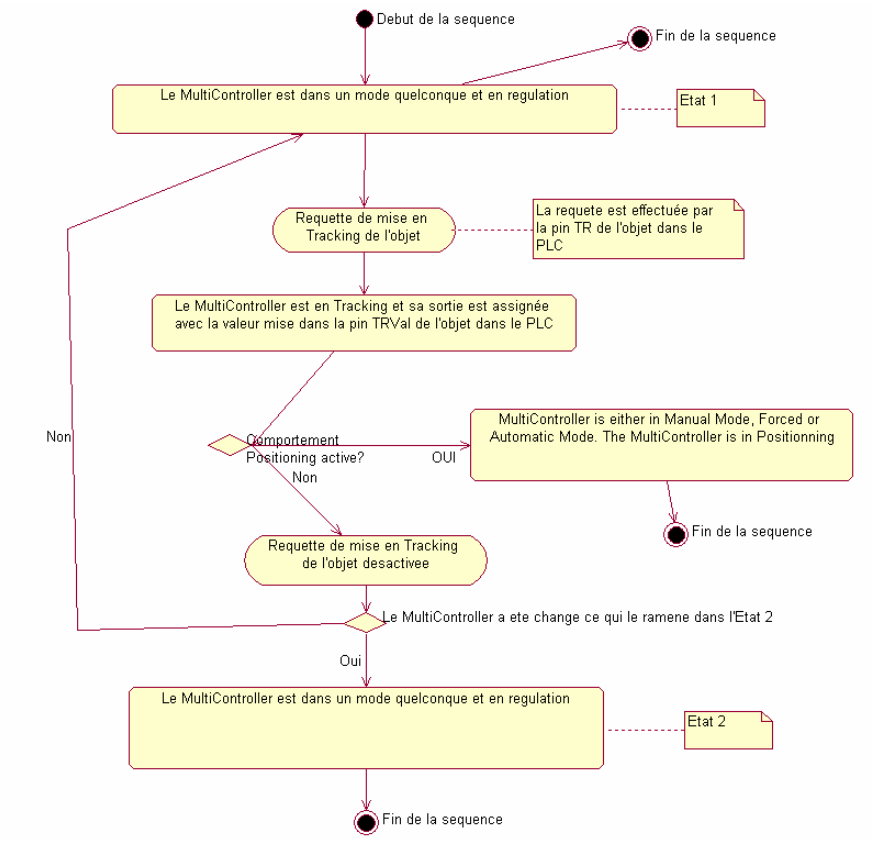


Figure 85 – Le fonctionnement du Tracking dans le MultiController

3.2. Design du MultiController

Voir annexe E – MultiController dans l'automate Schneider (UnityV2.1) – objets DFB.

3.2.1. Structure et organisation du code automate dans le DFB

La structure interne du code automate que j'ai introduite, se décompose de manière à permettre une facilité de développement. L'idée de base de l'organisation interne est de réaliser des 'zones' dédiées pour les fonctionnalités. Le découpage est celui-ci :

- Une zone pour la mise en place des modes, des comportements, des limitations, de la détermination sur l'état et les valeurs de l'objet.
- Une zone qui fait l'assignation des variables à la bonne régulation.
- Des zones qui sont réservées pour les algorithmes avancés.
- Une zone qui s'occupe de l'assignement des sorties avec la bonne régulation.
- Une dernière zone pour l'assignement de certains registres qui ne pouvaient se faire dans les zones précédentes.

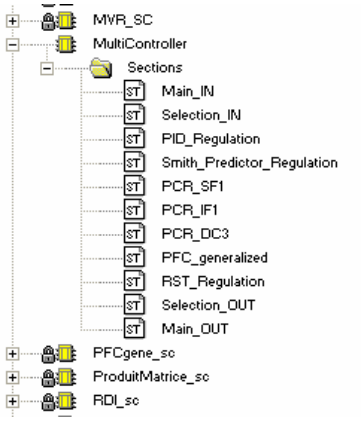


Figure 86 – Organisation du code Unity du MultiController pour les automates Schneider

La seconde caractéristique judicieuse du code automate du MultiController réside dans sa capacité d'encapsulation d'autres objets. Par cette fonctionnalité intéressante, j'ai donc encapsulé les algorithmes de régulations que j'ai développés auparavant. J'ai également pu ajouter certains objets protégés de Schneider (bibliothèque PCR).

Le principe est simple et repose sur la réutilisation dans un DFB d'autres DFB développés. Ainsi, un objet de commande avancée peut être réalisé 'en dehors' du MultiController mais être utilisé par un appel.

Ce principe d'encapsulation est une solution efficace pour travailler sur les codes dits de 'régulations pures' sans affecter les fonctionnalités de l'objet. Par ailleurs, cela permet de bloquer le code source de certaines commandes complexes pour éviter toutes manipulations non désirées dans celles-ci.

La figure suivante montre le MultiController avec les DFB de régulations encapsulées dans la structure de l'objet :

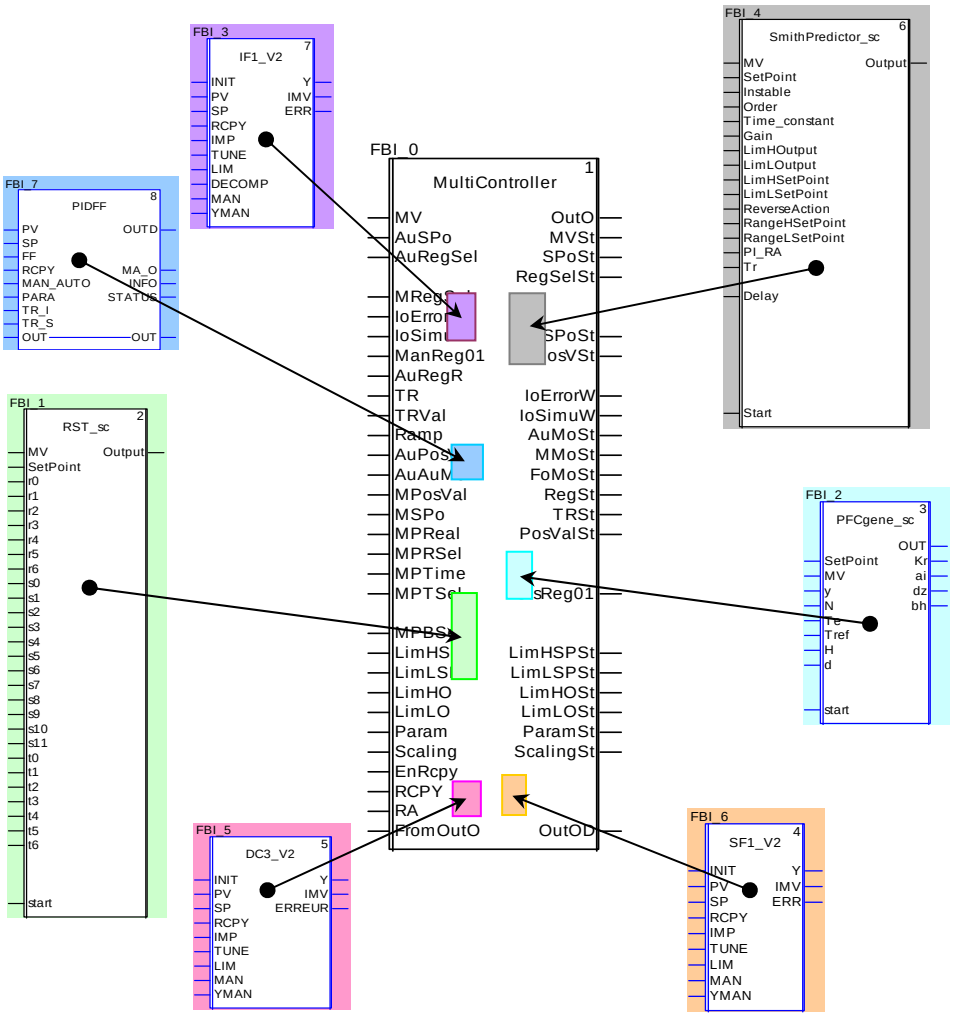


Figure 87 – Principe d'encapsulation des algorithmes de régulations dans le MultiController sous Unity

Le MultiController possède les algorithmes suivants :

- Le PID – bibliothèque classique Unity.
- Le Prédicteur de Smith pour systèmes stables (premier ordre et second ordre, pôle double) et intégrateurs – objet SmithPredictor_sc que j'ai développé.
- La commande prédictive fonctionnelle pour premier ordre – objet SF1_v2 de la bibliothèque PCR de Schneider.
- La commande prédictive fonctionnelle pour deuxième et troisième ordre – objet DC3_v2 de la bibliothèque PCR de Schneider.
- La commande prédictive fonctionnelle pour intégrateur – objet IF1_v2 de la bibliothèque PCR de Schneider.
- La commande prédictive fonctionnelle généralisée – objet PFCgene_sc que j'ai développé.
- Le contrôleur RST – objet RST_sc que j'ai développé (cet objet est également utilisable avec l'objet GPC_sc en dehors pour la commande GPC).

3.2.2. Interfaces utilisateur

L'interface utilisateur associée au MultiController dans le SCADA PVSS répond à plusieurs objectifs :

- Donner à l'utilisateur une interface avec les mêmes principes que ceux des objets du framework UNICOS. Ainsi, le MultiController n'est pas perçu comme une fonctionnalité complexe mais comme un objet UNICOS.
- Permettre à l'ingénieur de process gaz d'utiliser des algorithmes de régulations autres que le PID, de les choisir, de les expérimenter et de quantifier leurs efficacités.
- Donner des mécanismes simples d'utilisation des régulations durant l'opération.
- Assurer une notion de 'tests' et de tuning grâce aux recettes et à leurs sauvegardes.
- Familiariser les utilisateurs de l'objet aux techniques de régulations et aux méthodologies de mise en œuvre.

Les vues PVSS de la représentation HMI du MultiController sont de trois ordres :

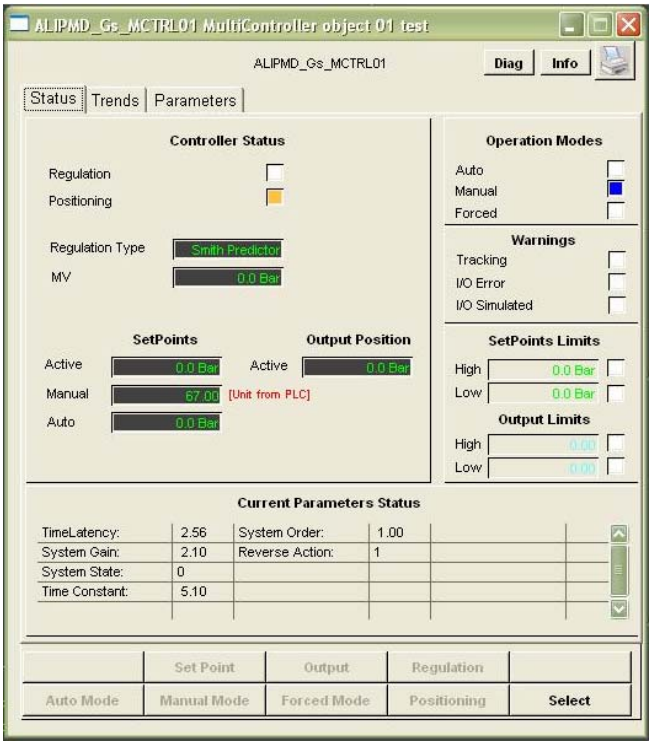


Figure 88 – Vue principale (Status) de l'interface du MultiController dans le SCADA PVSS

La vue principale concerne l'état général de l'objet avec ses modes et comportements (Status). Elle montre l'état des consignes et des valeurs mesurées ainsi que les limites. Enfin, elle permet de visualiser la régulation active avec les paramètres correspondants.

Par exemple, dans la figure précédente on peut voir que la régulation sélectionnée est un Prédicteur de Smith. La consigne automatique est nulle et la consigne manuelle est de 67. L'objet est en mode manuel et en comportement de positionnement. Les paramètres de la régulation sont de 5,1 secondes pour la constante de temps, d'un retard de 2.56 secondes, d'un gain de 2.1, et d'un premier ordre avec une action inversée pour le régulateur.

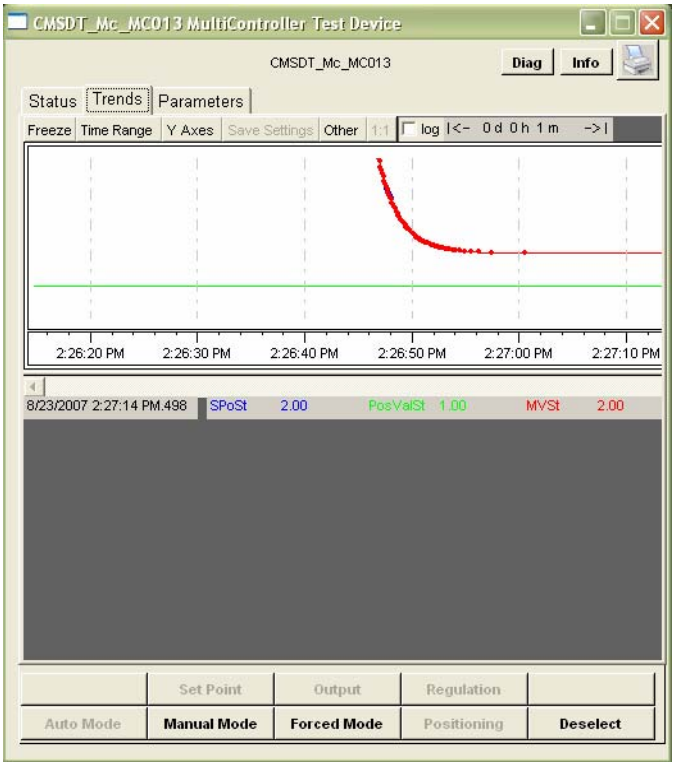


Figure 89 - Vue des courbes de tendances (Trends) de l'interface du MultiController dans le SCADA PVSS

La figure ci-dessus représente la vue PVSS des courbes de tendances. Elle permet de visualiser la valeur mesurée, la consigne et la commande du régulateur. Cette vue est un standard UNICOS développé avec les fonctionnalités PVSS classiques. Les courbes de tendances peuvent être configurables dans le temps, dans les échelles et dans les visualisations.

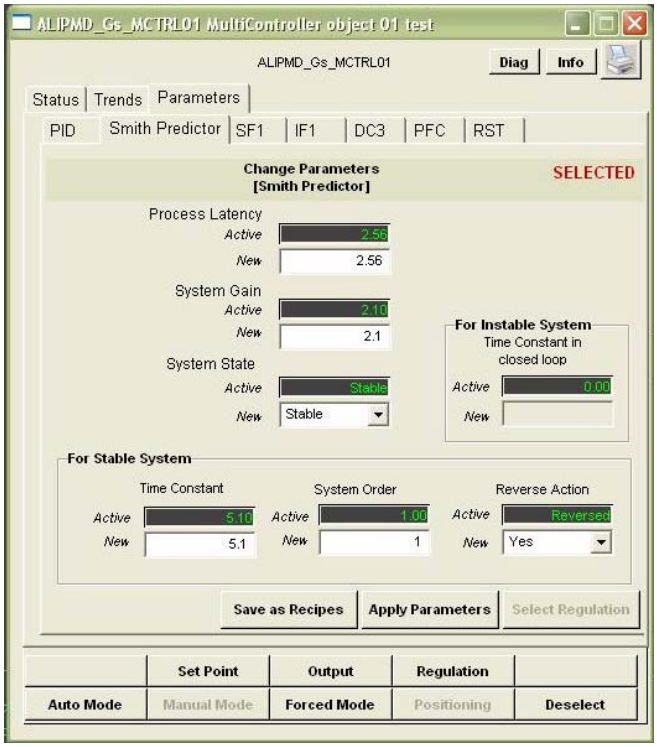


Figure 90 - Vue du panneau des paramètres de l'interface du MultiController dans le SCADA PVSS

Cette dernière vue représente les régulations disponibles dans l'objet. Chaque onglet (le Prédicteur de Smith dans l'exemple ci-dessus) permet de sélectionner la régulation désirée, et permet de changer les paramètres en ligne (uniquement en mode manuel).

Chaque onglet de régulation est muni d'une fonction de sauvegarde des paramètres manuels dans des recettes, (bouton 'Save as Recipe' dans la figure précédente). Les recettes créées peuvent ensuite être chargées dans la MultiController pour être appliquées en mode automatique.

3.3. Utilisation du MultiController dans l'automate

Dans cette partie, je montre les deux manières d'utiliser le MultiController dans l'automate programmable Premium sous Unity. J'expose deux exemples dans deux langages différents.

3.3.1. Langage par blocs fonctionnels (FBD)

Le langage FBD est un moyen simple d'instancier un MultiController. Il est ainsi possible de suivre en ligne les évolutions des variables, de voir le chemin des informations et d'avoir une vue globale de la régulation.

Le langage FBD se caractérise par une programmation graphique. Les instances sont directement identifiables et leurs relations clairement visibles.

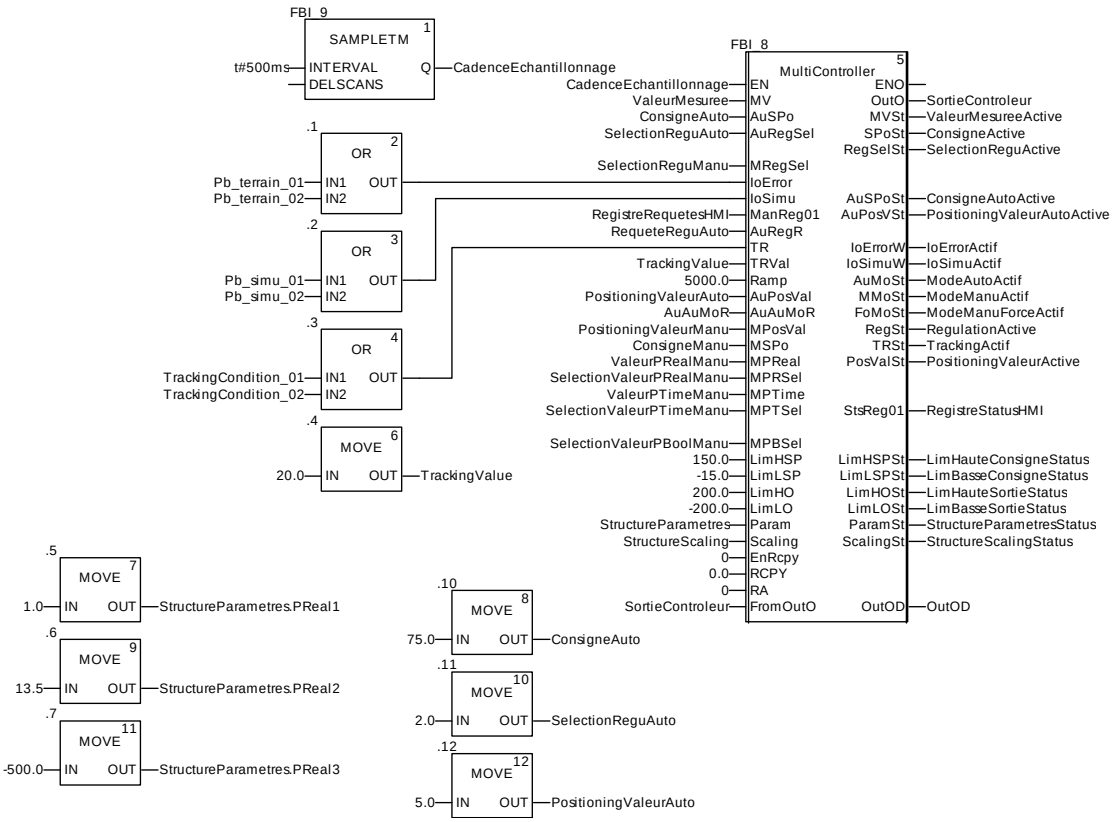


Figure 91 – Programmation simple en langage FBD (Unity) d'une régulation avec le MultiController

L'exemple de programmation FBD présenté-ci dessus met en évidence les interactions et les paramétrages. Grâce à cette programmation graphique, le diagnostic en ligne est plus lisible. En effet, lorsque l'automate est connecté à la console Unity du programme, les valeurs sont animées en temps réel.

Cette programmation peut connaître des limitations. Lorsque les stratégies deviennent plus complexes (démarrages, transitions entre deux états, etc.), la mise en œuvre avec des blocs fonctionnels est plus difficile. Le langage structuré peut alors prendre le relais pour faciliter l'implémentation.

3.3.2. Langage Structuré (ST)

Contrairement au langage FBD, le langage structuré n'est pas aussi convivial. Il a cependant le grand avantage de pouvoir mettre en œuvre des conditions plus complexes de démarrage et d'arrêt de la régulation (ou bien avec le Tracking). Par ailleurs, ce langage est beaucoup plus adapté pour le framework GCS. Le développement est fait en amont d'Unity pour importer le code final en ST.

```
(*instanciation du MultiController*)
ALIPMD_Gs_MCTRL01 (
(*entrees *)
EN:=ALIPMD_Gs_MCTRL01_EN, MV := ALIPMD_Gs_MCTRL01_MV, AuSPo := ALIPMD_Gs_MCTRL01_AuSPo,AuRegSel := ALIPMD_Gs_MCTRL01_AuRegSel,
MRegSel := ALIPMD_Gs_MCTRL01_MRegSel, IoError := ALIPMD_Gs_MCTRL01_IoError, IoSimu := ALIPMD_Gs_MCTRL01_IoSimu,
ManReg01 := ALIPMD_Gs_MCTRL01_ManReg01, AuRegR := ALIPMD_Gs_MCTRL01_AuRegR, TR := ALIPMD_Gs_MCTRL01_TR,
TRVal := ALIPMD_Gs_MCTRL01_TRVal, Ramp := ALIPMD_Gs_MCTRL01_Ramp, AuPosVal := ALIPMD_Gs_MCTRL01_AuPosVal,
AuAuMoR := ALIPMD_Gs_MCTRL01_AuAuMoR, MPosVal := ALIPMD_Gs_MCTRL01_MPosVal,MSPo := ALIPMD_Gs_MCTRL01_MSPo,
MPReal := ALIPMD_Gs_MCTRL01_MPReal,MPRSel := ALIPMD_Gs_MCTRL01_MPRSel, MPTime := ALIPMD_Gs_MCTRL01_MPTime,
MPTSel:= ALIPMD_Gs_MCTRL01_MPTSel, MPBSel := ALIPMD_Gs_MCTRL01_MPBSel, LimHSP := ALIPMD_Gs_MCTRL01_LimHSP,
LimLSP:= ALIPMD_Gs_MCTRL01_LimLSP, LimHO := ALIPMD_Gs_MCTRL01_LimHO,LimLO := ALIPMD_Gs_MCTRL01_LimLO,
Param := ALIPMD_Gs_MCTRL01_Param,Scaling := ALIPMD_Gs_MCTRL01_Scaling, EnRcpy:= ALIPMD_Gs_MCTRL01_EnRcpy,
RCPY := ALIPMD_Gs_MCTRL01_RCPY, RA := ALIPMD_Gs_MCTRL01_RA,FromOutO := ALIPMD_Gs_MCTRL01_FromOutO,
(* sorties *)
OutO => ALIPMD_Gs_MCTRL01_OutO,MVSt => ALIPMD_Gs_MCTRL01_MVSt, SPoSt => ALIPMD_Gs_MCTRL01_SPoSt,
RegSelSt => ALIPMD_Gs_MCTRL01_RegSelSt, AuSPoSt => ALIPMD_Gs_MCTRL01_AuSPoSt,AuPosVSt => ALIPMD_Gs_MCTRL01_AuPosVSt,
PosValSt => ALIPMD_Gs_MCTRL01_PosValSt,StsReg01 => ALIPMD_Gs_MCTRL01_StsReg01, LimHSPSt => ALIPMD_Gs_MCTRL01_LimHSPSt,
LimLSPSt => ALIPMD_Gs_MCTRL01_LimLSPSt, LimHOSt => ALIPMD_Gs_MCTRL01_LimHOSt,LimLOSt => ALIPMD_Gs_MCTRL01_LimLOSt,
ParamSt => ALIPMD_Gs_MCTRL01_ParamSt);

(*Echantillonnage*)
SAMPLETM (INTERVAL:=t#100ms, Q=>Q_Enable);
ALIPMD_Gs_MCTRL01_EN:=Q_Enable;

ALIPMD_Gs_MCTRL01_MV:=ALIPMD_Di_PT6124_PosSt;          ALIPMD_Gs_MCTRL01_AuSPo:=ALIPMD_Gs_M01SetPoint_CurValSt;
ALIPMD_Gs_MCTRL01_AuRegSel:=ALIPMD_Gs_M01RegSel_CurValSt;          ALIPMD_Gs_MCTRL01_TRVal:=33.5;
ALIPMD_Gs_MCTRL01_Ramp:=250.0;     ALIPMD_Gs_MCTRL01_AuPosVal:=ALIPMD_Gs_M01Position_CurValSt;
ALIPMD_Gs_MCTRL01_LimHSP:=ALIPMD_Gs_M01LimHSP_CurValSt;          ALIPMD_Gs_MCTRL01_LimLSP:=ALIPMD_Gs_M01LimLSP_CurValSt;
ALIPMD_Gs_MCTRL01_LimHO:=ALIPMD_Gs_M01LimHO_CurValSt;          ALIPMD_Gs_MCTRL01_LimLO:=ALIPMD_Gs_M01LimLO_CurValSt;
ALIPMD_Gs_MCTRL01_RA:=ALIPMD_Gs_M01RA_CurVal;          ALIPMD_Gs_MCTRL01_FromOutO:=ALIPMD_Gs_MCTRL01_OutO;

(*Conditions de mise en Regulation*)
IF ALIPMD_Gs_GsPCO.RunO
THEN          ALIPMD_Gs_MCTRL01_AuRegR:=TRUE;
ELSE          ALIPMD_Gs_MCTRL01_AuRegR:=FALSE;
END_IF;

(*IoError conditions*)
IF ALIPMD_Gs_PT0101_PosSt > 1500.0
THEN          ALIPMD_Gs_MCTRL01_IoError:=TRUE;
ELSE          ALIPMD_Gs_MCTRL01_IoError:=FALSE;
END_IF;

(*IoSimu conditions*)
IF ALIPMD_Gs_PT0101_PosSt < 1.0
THEN          ALIPMD_Gs_MCTRL01_IoSimu:=TRUE;
ELSE          ALIPMD_Gs_MCTRL01_IoSimu:=FALSE;
END_IF;

(*Conditions de mise en Tracking*)
IF ALIPMD_Gs_PT0101_PosSt = 1234.5
THEN          ALIPMD_Gs_MCTRL01_TR:=TRUE;
ELSE          ALIPMD_Gs_MCTRL01_TR:=FALSE;
END_IF;

(*Assignation des recettes aux parametres*)
ALIPMD_Gs_MCTRL01_Param.PReal1:=ALIPMD_Gs_M01PReal01_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal2:=ALIPMD_Gs_M01PReal02_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal3:=ALIPMD_Gs_M01PReal03_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal4:=ALIPMD_Gs_M01PReal04_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal5:=ALIPMD_Gs_M01PReal05_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal6:=ALIPMD_Gs_M01PReal06_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal7:=ALIPMD_Gs_M01PReal07_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal8:=ALIPMD_Gs_M01PReal08_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal9:=ALIPMD_Gs_M01PReal09_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal10:=ALIPMD_Gs_M01PReal10_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal11:=ALIPMD_Gs_M01PReal11_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal12:=ALIPMD_Gs_M01PReal12_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal13:=ALIPMD_Gs_M01PReal13_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal14:=ALIPMD_Gs_M01PReal14_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal15:=ALIPMD_Gs_M01PReal15_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal16:=ALIPMD_Gs_M01PReal16_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal17:=ALIPMD_Gs_M01PReal17_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal18:=ALIPMD_Gs_M01PReal18_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal19:=ALIPMD_Gs_M01PReal19_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PReal20:=ALIPMD_Gs_M01PReal20_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PReal21:=ALIPMD_Gs_M01PReal21_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PTime1:=ALIPMD_Gs_M01PTime01_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PTime2:=ALIPMD_Gs_M01PTime02_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PTime3:=ALIPMD_Gs_M01PTime03_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PTime4:=ALIPMD_Gs_M01PTime04_CurValSt;          ALIPMD_Gs_MCTRL01_Param.PTime5:=ALIPMD_Gs_M01PTime05_CurValSt;
ALIPMD_Gs_MCTRL01_Param.PBool1:=ALIPMD_Gs_M01PBool01_CurVal;          ALIPMD_Gs_MCTRL01_Param.PBool2:=ALIPMD_Gs_M01PBool02_CurVal;
ALIPMD_Gs_MCTRL01_Param.PBool3:=ALIPMD_Gs_M01PBool03_CurVal;          ALIPMD_Gs_MCTRL01_Param.PBool4:=ALIPMD_Gs_M01PBool04_CurVal;
ALIPMD_Gs_MCTRL01_Param.PBool5:=ALIPMD_Gs_M01PBool05_CurVal;
```

Figure 92 – Programmation simple en langage Structuré ST (Unity) d'une régulation avec le MultiController

L'exemple précédent illustre la manière de programmer le MultiController en texte structuré. Ce langage permet une relative liberté d'implémentation et peut assurer des stratégies complexes de programmation.

L'avantage du langage est le traitement des instructions dans l'ordre dans lequel elles sont écrites. A chaque cycle de l'automate, on repasse invariablement au travers du programme suivant l'ordre du code implémenté. Ceci est à mettre en comparaison avec le langage FBD qui exécute les instructions bloc par bloc, (l'ordre d'exécution suit l'ordre des instances crées dans le programme).

3.4. Le futur avec le MultiController

Actuellement, le MultiController est opérationnel. La manière dont j'ai conçu le Design de l'objet, autorise les utilisateurs à travailler avec prés de huit algorithmes différents de régulations.

Le kit d'outils d'Automatique sous automate Schneider permet de réaliser des solutions plus élaborées de boucles de régulations (Voir annexe A). Cela laisse entrevoir des possibilités supplémentaires pour les utilisateurs du MultiController. Pour le moment, les objets DFB de modélisation/identification ne sont pas intégrés en tant que tel au framework GCS. Ils ne sont pas représentés dans le HMI, et ne sont donc pas 'contrôlables' en ce sens. Ils peuvent cependant être utilisés ponctuellement avec le MultiController dans l'automate programmable.

L'exemple suivant montre une utilisation du MultiController pour la mise en place d'une régulation adaptive avec la GPC.

Régulation GPC adaptive

Soit le système simulé H(p) à corriger : $H(p) = \frac{2}{1+10p}$

Le contrôle adaptatif réalisé par le MultiController, utilise l'objet GPC_sc (commande prédictive généralisée) et l'objet MCR_sc (moindres carrés récursifs) pour la détermination en ligne du modèle à corriger.

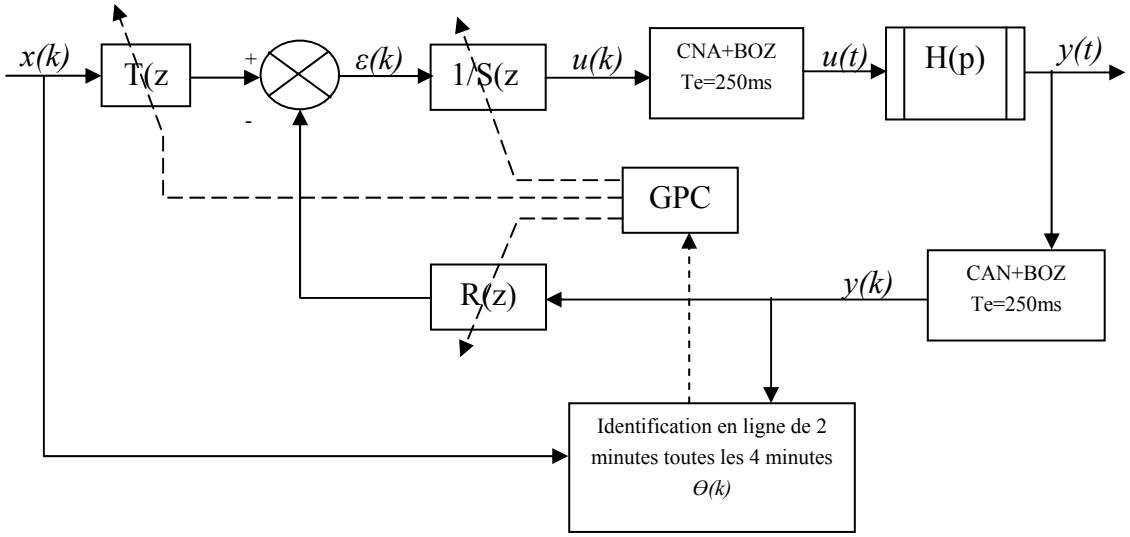


Figure 93 – Exemple d'utilisation du MultiController : contrôle adaptatif dans l'automate programmable Schneider – principe

Le principe du contrôle adaptatif de cet exemple est simple :

- Le MultiController est utilisé avec le contrôleur RST. L'objet GPC_sc est directement connecté à la structure des paramètres. L'adaptation est effectuée grâce à l'objet MCR_sc qui identifie en ligne le modèle du process à corriger H. Le bloc SBPAfix est utilisé pour générer des échelons suffisants pour permettre une identification convenable. Pour des raisons de convergence et d'initialisation de l'algorithme des moindres carrés récursifs du bloc, l'identification s'effectue cycliquement toutes les quatre minutes pendant 2 minutes.
- Les paramètres de la commande prédictive généralisée sont :
  - N_1=1 car pas de délais
  - N_2=5 pour permettre un horizon suffisant
  - N_u=1 (valeur par défaut)
  - λ~0.013 qui est le calcul optimal en ligne
  - λ_1=0.999 et λ_2=1pour avoir un léger facteur d'oubli est un gain décroissant.
  - F_0=10.0 pour permettre une identification qui converge convenablement
- Au démarrage, le système fait une identification paramétrique de deux minutes avant de commencer la régulation. (le Bloc SBPAfix_sc génère des échelons en boucle ouverte). Lorsque la régulation est démarrée, le contrôle adaptif s'effectue en boucle fermée toutes les quatre minutes. Le principe est que le contrôle travaille sur la consigne pour produire des variations suffisantes, (variation de 5 % avec le même bloc SBPAfix_sc).

La figure suivante montre le code automate sous Schneider de ce contrôle adaptatif :

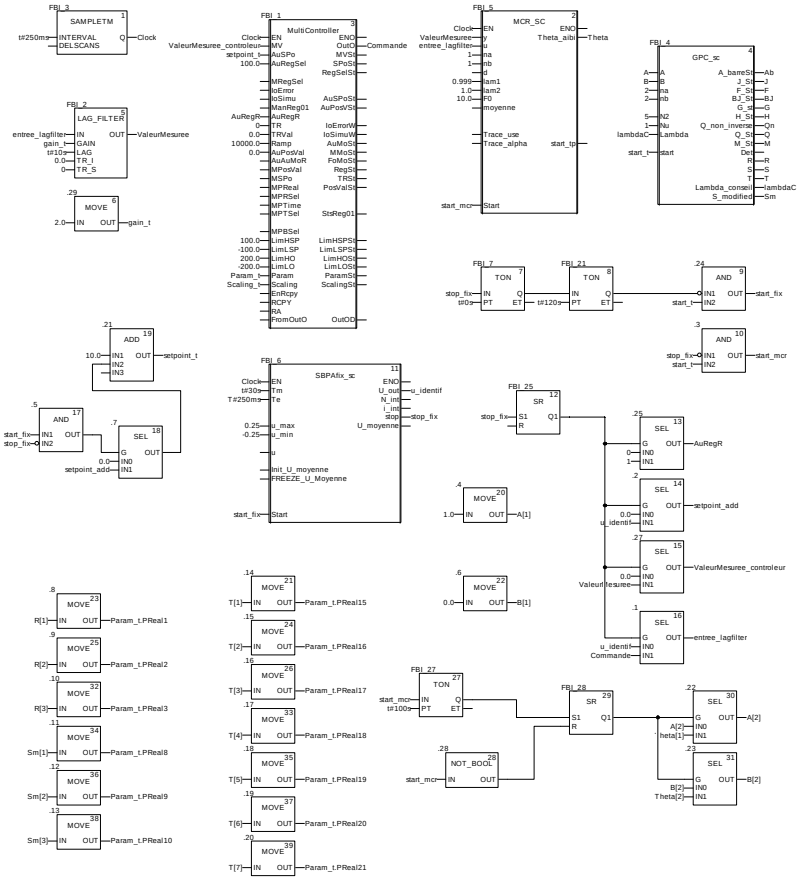


Figure 94 – Utilisation du MultiController : contrôle adaptatif dans l'automate programmable Schneider – Programme automate

Comme je l'ai explicité précédemment dans ce document, la programmation en langage DFB est plus complexe à réaliser. Elle permet cependant une meilleure visualisation des variables du système en ligne. Dans cet exemple, on peut entrevoir les divers liens entre le MultiController et les blocs du kit d'outils d'Automatique sous automate Schneider.

Toute la difficulté du programme mis en place est de bien organiser les blocs fonctions. Le code doit être suffisamment espacé avec peu de connections directes. A contrario, les blocs doivent êtres subtilement agencés pour permettre de voir à l'écran les valeurs pertinentes connexes en même temps. Ce contrôle adaptatif en est un exemple.

On obtient les résultats suivant sur la sortie et la commande :

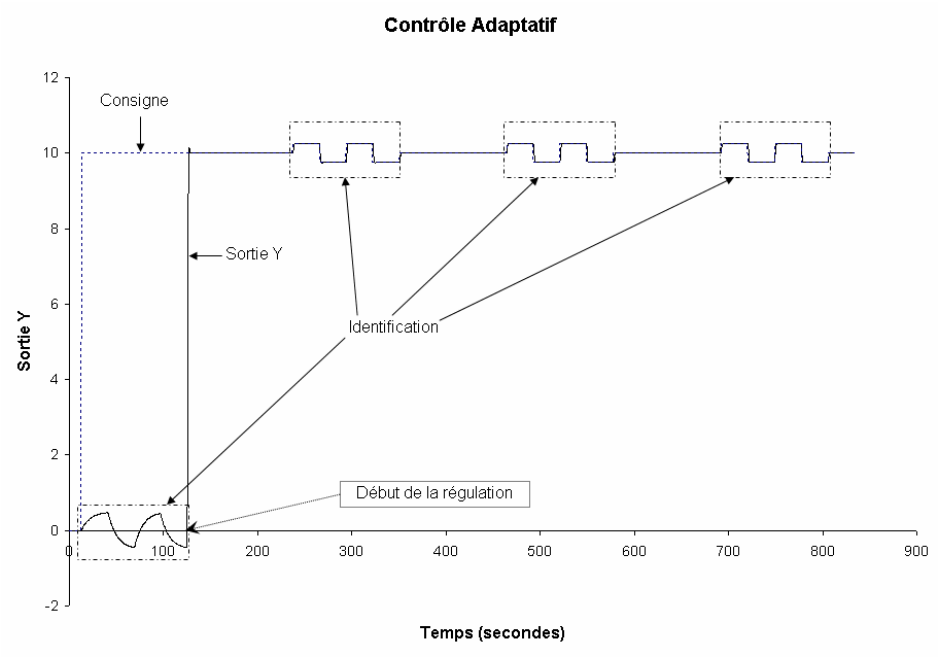


Figure 95 – Exemple d'utilisation du MultiController : contrôle adaptatif dans l'automate programmable Schneider – Sortie système

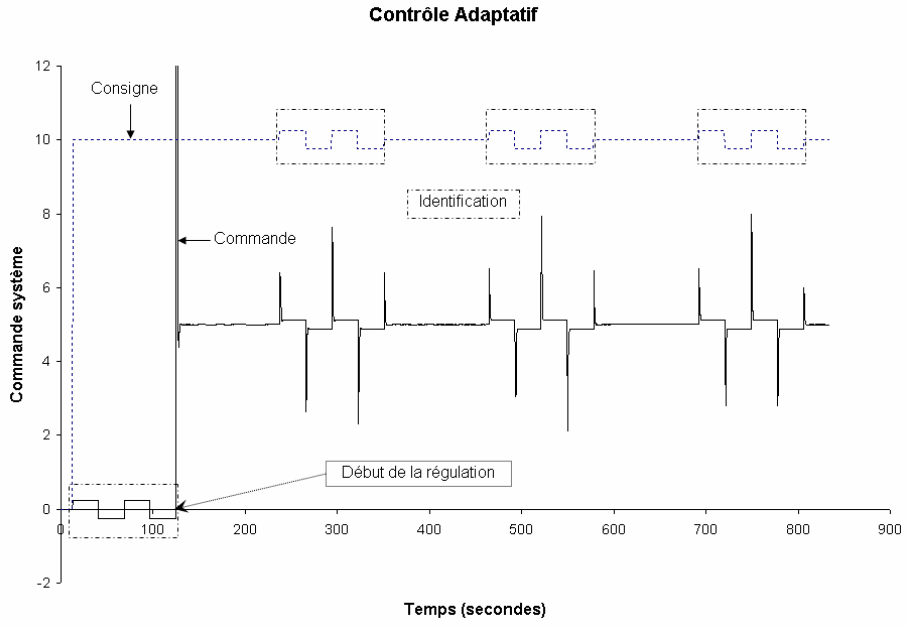


Figure 96 – Exemple d'utilisation du MultiController : contrôle adaptatif dans l'automate programmable Schneider – Commande système

Les figures montrent que la sortie du régulateur assure une correction convenable suite à la phase initiale d'identification. En effet, les paramètres proposés par l'algorithme GPC au RST semblent assurer une bonne correction du système. Enfin, les identifications successives sont parfaitement réalisées en boucle fermée.

Cet exemple illustre les possibilités futures offertes aux utilisateurs des systèmes de contrôle du projet GCS.

L'intégration dans le framework et le développement dirigé par les modèles du kit d'outils d'Automatique sous automate Schneider serait vraisemblablement l'étape suivante. Ces travaux seraient un parfait complément des avancées effectuées avec le MultiController.

4. Conclusion

Le MultiController que j'ai créé est une réalisation faisant la synthèse des besoins nécessaires en algorithmes de régulations pour le projet Gaz. Il prend en considération les stratégies nouvelles de développement pour s'intégrer pleinement dans le contexte. Cet 'objet' utilise également une partie du savoir faire du CERN (framework UNICOS) pour rétribuer toutes les fonctionnalités des algorithmes aux utilisateurs. Incorporé au nouveau framework du projet GCS, le MultiController permet de systématiser l'implémentation d'algorithmes de contrôles avancés sur l'ensemble des 23 installations à gaz. Enfin, le MultiController offre aux utilisateurs un standard d'utilisation par le biais du 'traitement objet' dans l'automate et dans le SCADA.

Conclusion

L'introduction d'algorithmes avancés de contrôle commande dans l'automate programmable industriel est une démarche nouvelle industrielle.

Les aspects que je présente dans la thèse mettent en lumière les possibilités offertes par l'automate. Les langages de programmation avec la norme CEI 61131-3 assurent des perspectives d'intégration très intéressantes. L'identification en ligne, les tests de validation et les algorithmes de régulations trouvent ainsi un cadre applicatif très large. L'automate n'est plus perçu comme une boite noire assurant les séquences et les modes de comportement du process. Il peut servir à l'asservissement de systèmes complexes et peut aussi être un organe de contrôle adaptatif.

Les travaux que j'ai présentés montrent la qualité de la synthèse d'algorithmes de régulations avancées dans un outil industriel. Ils démontrent également qu'il est tout à fait envisageable d'intégrer des algorithmes élaborés dans un projet à grande échelle.

L'étude de nouvelles orientations de développement du projet Gaz du CERN pour la réalisation des travaux de thèse était nécessaire. Par l'étude des méthodologies framework et Model Driven, j'ai entrepris une réflexion supplémentaire sur l'intégration des algorithmes avancés.

Le MultiController que j'ai développé est une synthèse des exigences du projet gaz qui utilise deux approches de développement du génie logiciel. En ce sens, des algorithmes prédictifs de contrôle peuvent être incorporés dans un projet innovant tout en répondant aux besoins de base (de faire) de la régulation. Les utilisateurs ont donc à disposition des outils de régulations avancées tout en conservant leur environnement propre. Le MultiController standardise l'utilisation des algorithmes avancés grâce à une ergonomie identique proposée au travers du SCADA. Enfin, cet objet systématise l'implémentation des régulations dans le projet GCS.

Les résultats développés dans cette thèse ouvrent des perspectives nouvelles sur le transfert de connaissances sur des plateformes industrielles à base d'automates programmables. Les chaînes de productions peuvent ainsi être pourvues de nouvelles fonctionnalités de régulation non envisagées dans le passé.

L'application de la « théorie papier » est maintenant sur le point d'être accomplie sur le mode industriel. Les limitations sont moins contraignantes et la recherche appliquée pour les solutions industrielles de contrôle des commandes ont encore de beaux enjeux à relever.

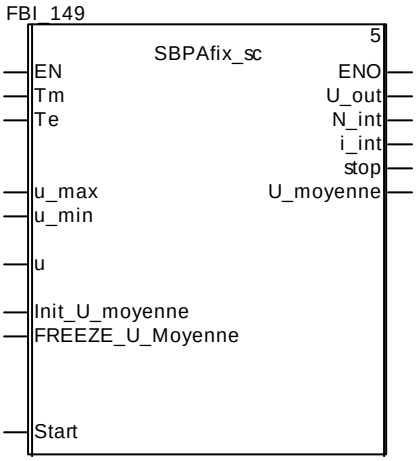
Annexes

Annexe A – Le kit d'outils d'Automatique sous automate Schneider (UnityV2.1) – objets DFB

SBPAfix_sc

Description

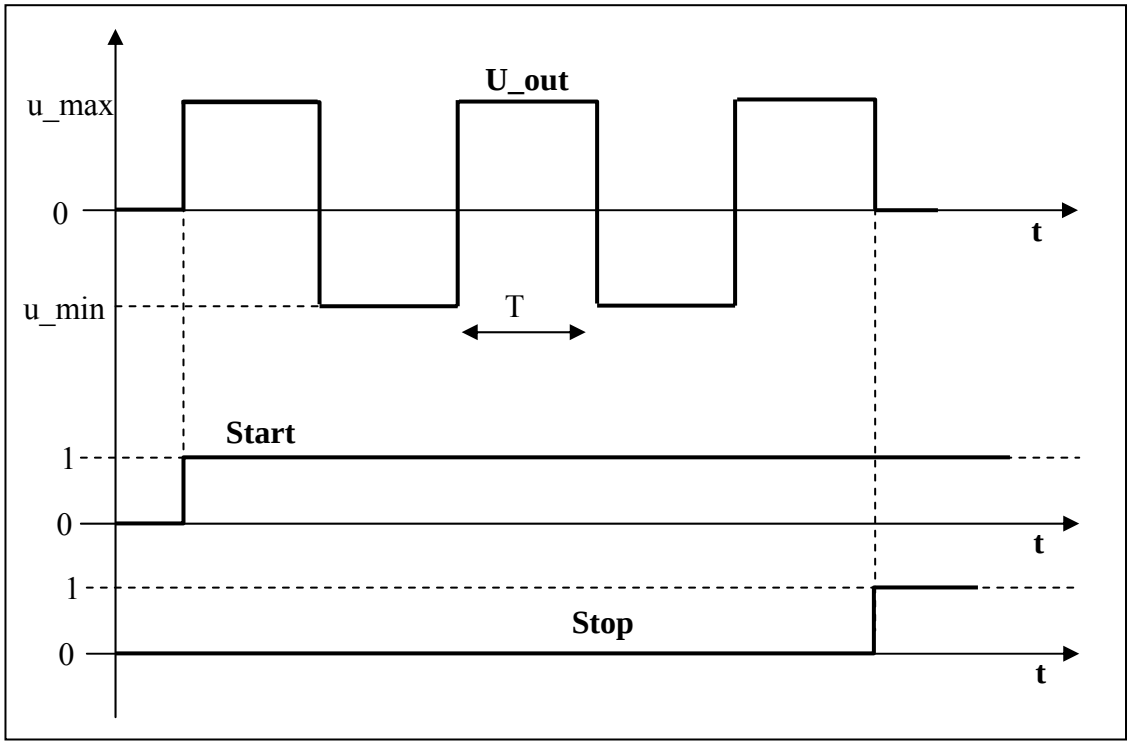
Objet permettant de générer trois échelons successifs. Il est également utilisé pour faire le calcul de la moyenne des entrées appliquées au système à identifier.



Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
Tm	TIME	Temps supérieur à 95% de la valeur finale du système soumis à un échelon
Te	TIME	Echantillonnage
u_max	REAL	Limite haute de l'échelon
u_min	REAL	Limite basse de l'échelon
u	REAL	Entrée (recopie) de l'excitation soumise au système à identifier
Init_U_moyenne	BOOL	Initialise le calcul de l'excitation u moyenne
FREEZE_U_Moyenne	BOOL	Bloque le calcul de u moyenne
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
U_out	REAL	Sortie à appliquer au système à identifier
N_int	INTEGER	N est le calcul du nombre de coup d'échantillonnages nécessaires pour chaque échelon (N=Tm/Te)

i_int	INTEGER	Compteur indicatif du nombre d'échantillons
Stop	BOOL	Sortie mise à 1 lorsque la séquence des 3 échelons est terminée
U_moyenne	REAL	U moyenne calculée (permet d'être soustraite à u pour l'identification)

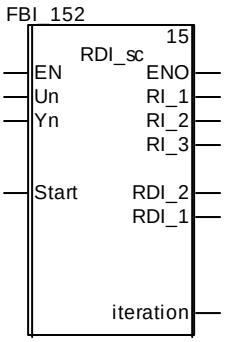
Principe de fonctionnement :



RDI_sc

Description

Objet permettant de faire le test en ligne des rapports des déterminants et des résidus instrumentaux.

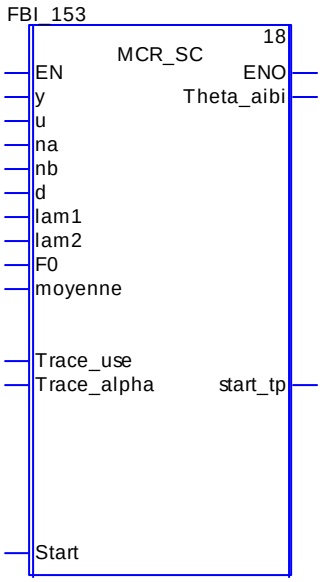


Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
Un	REAL	Entrée du process
Yn	REAL	Sortie du process
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
RI_1	REAL	RI(1)
RI_2	REAL	RI(2)
RI_3	REAL	RI(3)
RDI_1	REAL	RDI(1)
RDI_2	REAL	RDI(2)
iteration	INTEGER	Compteur pour le nombre d'itérations

MCR_sc

Description

Objet permettant de faire une identification paramétrique en ligne les moindres carrés récursifs.



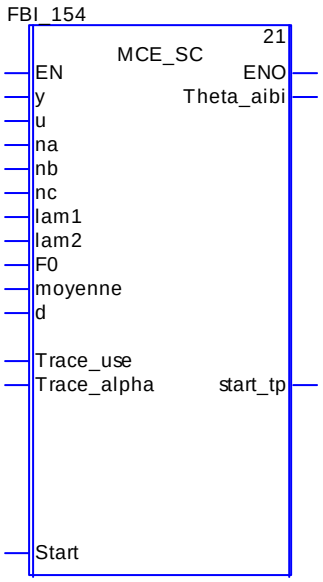
Entrées	Format	Description
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
y	REAL	Sortie du process
u	REAL	Entrée su process
na	INTEGER	Nombre de coefficients pour A
nb	INTEGER	Nombre de coefficients pour B
d	INTEGER	Délai (multiple de Te)
Lam1	REAL	Lambda 1 dans la méthode (<=1, et généralement > 0.95)
Lam2	REAL	Lambda 2 dans la méthode (<=1, et généralement > 0.95)
F0	REAL	Initialisation de la matrice F par F0 (éléments de la diagonale)
moyenne	REAL	Valeur moyenne (si existante) a soustraire de l'entrée u.
Trace_use	BOOL	Utilisation de la trace de la matrice F (trace constante)
Trace_alpha	REAL	Poids lors de l'utilisation de la trace constante pour la matrice F

Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
Theta_aibi	ARRAY[50 REALS]	Tableau du vecteur des paramètres identifiés
Start_tp	BOOL	Indique le démarrage et l'arrêt de l'identification

MCE_sc

Description

Objet permettant de faire une identification paramétrique en ligne les moindres carrés étendus.



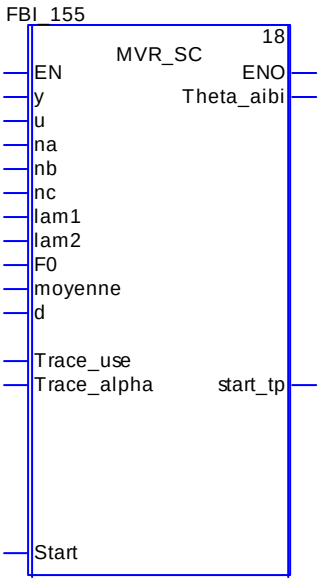
Entrées	Format	Description
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
y	REAL	Sortie du process
u	REAL	Entrée su process
na	INTEGER	Nombre de coefficients pour A
nb	INTEGER	Nombre de coefficients pour B
nc	INTEGER	Nombre de coefficients pour C
Lam1	REAL	Lambda 1 dans la méthode (<=1, et généralement > 0.95)
Lam2	REAL	Lambda 2 dans la méthode (<=1, et généralement > 0.95)
F0	REAL	Initialisation de la matrice F par F0 (éléments de la diagonale)
moyenne	REAL	Valeur moyenne (si existante) a soustraire de l'entrée u.
d	INTEGER	Délai (multiple de Te)
Trace_use	BOOL	Utilisation de la trace de la matrice F (trace constante)
Trace_alpha	REAL	Poids lors de l'utilisation de la trace constante pour

		la matrice F
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
Theta_aibi	ARRAY[50 REALS]	Tableau du vecteur des paramètres identifiés
Start_tp	BOOL	Indique le démarrage et l'arrêt de l'identification

MVR_sc

Description

Objet permettant de faire une identification paramétrique en ligne le maximum de vraisemblance récursif.



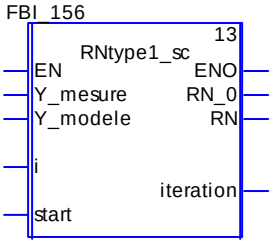
Entrées	Format	Description
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
y	REAL	Sortie du process
u	REAL	Entrée su process
na	INTEGER	Nombre de coefficients pour A
nb	INTEGER	Nombre de coefficients pour B
nc	INTEGER	Nombre de coefficients pour C
Lam1	REAL	Lambda 1 dans la méthode (<=1, et généralement > 0.95)
Lam2	REAL	Lambda 2 dans la méthode (<=1, et généralement > 0.95)
F0	REAL	Initialisation de la matrice F par F0 (éléments de la diagonale)
moyenne	REAL	Valeur moyenne (si existante) a soustraire de l'entrée u.
d	INTEGER	Délai (multiple de Te)
Trace_use	BOOL	Utilisation de la trace de la matrice F (trace constante)
Trace_alpha	REAL	Poids lors de l'utilisation de la trace constante pour

		la matrice F
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
Theta_aibi	ARRAY[50 REALS]	Tableau du vecteur des paramètres identifiés
Start_tp	BOOL	Indique le démarrage et l'arrêt de l'identification

RNType1_sc

Description

Objet permettant de faire le test de blanchissement des erreurs de prédiction en ligne aux sens des moindres carrée (méthode valide pour les MCR, MCE et MVR).

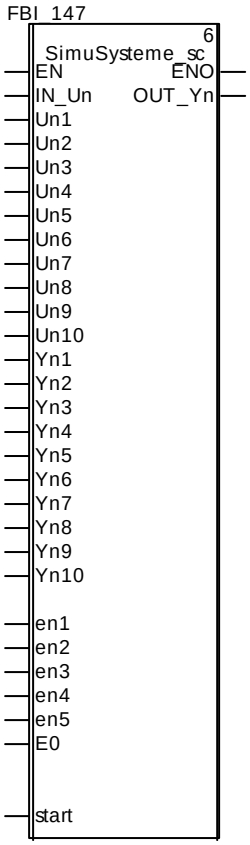


Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
Y_mesure	REAL	Sortie réelle du process
Y_modele	REAL	Sortie du modèle identifié soumis aux mêmes entrées que le système réel.
i	INTEGER	Nombre de RN(i) à calculer
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
RN_0	REAL	RN(0)
RN	ARRAY[50 REALS]	RN(i)
Start_tp	BOOL	Indique le démarrage et l'arrêt de l'identification

SimuSysteme_sc

Description

Objet permettant de simuler un système de type ARMA ou CARIMA.



Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
IN_Un	REAL	Entrée du modèle
Un1…Un10	REAL	Coefficients du polynôme B [Un2==coefficient de U(n-2)]
Yn1…Yn10	REAL	Coefficients du polynôme A [Yn2==coefficient de Y(n-2)]
en1…en5	REAL	Coefficients du polynôme C [en2==coefficient de e(n-2)]
E0	REAL	Entrée erreur processus/modèle (Yréel-Ymodèle)
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
OUT_Yn	REAL	Sortie du modèle

Annexe B – GPC- exemple d'une mise en œuvre pratique

Le Prédicteur est de la forme :

$$y(t+j) = \underbrace{F_j(q^{-1})y(t) + H_j(q^{-1})\Delta u(t-1)}_{passée} + \\ + \underbrace{G_j(q^{-1})\Delta u(t+j-1) + J_j(q^{-1})\xi(t)}_{futur}$$

Les équations diophantienne à résoudre sont :

$$\begin{cases} A(q^{-1})J_j(q^{-1})\Delta(q^{-1}) + q^{-j}F_j(q^{-1}) = 1 \\ B(q^{-1})J_j(q^{-1}) = G_j(q^{-1}) + q^{-j}H_j(q^{-1}) \end{cases}$$

La solution de ces équations est donnée par :

$$\begin{cases} J_{j+1}(q^{-1}) = J_j(q^{-1}) + F_j(0)q^{-j} \\ F_{j+1}(q^{-1}) = q\left[F_j(q^{-1}) - F_j(0).\overline{A}(q^{-1})\right] \\ \overline{A}(q^{-1}) = A(q^{-1}).\Delta(q^{-1}) \end{cases}$$

Avec les conditions initiales :

$$\begin{cases} J_1(q^{-1}) = 1 \\ F_1(q^{-1}) = q\left[1 - \overline{A}(q^{-1})\right] \\ \overline{A}(q^{-1}) = A(q^{-1}).\Delta(q^{-1}) \end{cases}$$

Et :

$$\begin{vmatrix} B(q^{-1})J_1(q^{-1}) = G_1(q^{-1}) + q^{-1}H_1(q^{-1}) \\ B(q^{-1})J_2(q^{-1}) = G_2(q^{-1}) + q^{-2}H_2(q^{-1}) \\ \qquad\qquad . \\ B(q^{-1})J_j(q^{-1}) = G_j(q^{-1}) + q^{-j}H_2(q^{-1}) \end{vmatrix}$$

Ensuite il suffit de calculer $M = Q.G^T$ avec $Q = \left[G^T.G + \lambda.I_{N_u}\right]^{-1}$ en tenant compte de la minimisation du cout quadratique et donc des paramètres N_1, N_2, N_u et λ. De cette matrice $M = Q.G^T$, on déduit directement m_1 qui nous permet de calculer la commande optimale.

Exemple

Soit le système tel que :

$$H(q^{-1}) = \frac{0.02q^{-1} + 0.01q^{-2}}{1 - 1.5q^{-1} + 0.8q^{-2}} = \frac{q^{-1}\left[0.02 + 0.01q^{-1}\right]}{1 - 1.5q^{-1} + 0.8q^{-2}} = \frac{q^{-1}B(q^{-1})}{A(q^{-1})}$$

On donne $N_1 = 1$, $N_2 = 3$, $N_u = 2$ et $\lambda = 0.2$

Le modèle CARIMA est donc :

$$\left[1 - 1.5q^{-1} + 0.8q^{-2}\right]y(t) = \left[0.02 + 0.01q^{-1}\right]u(t-1) + \frac{\xi(t)}{\Delta(q^{-1})}$$

On résout les équations diophantiennes avec pour commencer :

$$\begin{cases} J_1(q^{-1}) = 1 \\ F_1(q^{-1}) = q\left[1 - \overline{A}(q^{-1})\right] \\ \overline{A}(q^{-1}) = A(q^{-1}).\Delta(q^{-1}) \end{cases}$$

On obtient :

$$\begin{cases} J_1(q^{-1}) = 1 \\ F_1(q^{-1}) = q\left[1 - \left(1 - 1.5q^{-1} + 0.8q^{-2}\right) * \left(1 - q^{-1}\right)\right] \\ F_1(q^{-1}) = 2.5 - 2.3q^{-1} + 0.8q^{-2} \\ \overline{A}(q^{-1}) = 1 - 2.5q^{-1} + 2.3q^{-2} - 0.8q^{-3} \end{cases}$$

Par itération successive on obtient les relations de précédentes :

$$\begin{cases} J_2(q^{-1}) = J_1(q^{-1}) + F_1(0)q^{-1} = 1 + 2.5q^{-1} \\ F_2(q^{-1}) = q\left[F_1(q^{-1}) - F_1(0).\overline{A}(q^{-1})\right] = q\left[2.5 - 2.3q^{-1} + 0.8q^{-2} - 2.5 * \left(1 - 2.5q^{-1} + 2.3q^{-2} - 0.8q^{-3}\right)\right] \\ F_2(q^{-1}) = 3.95 - 4.95q^{-1} + 2q^{-2} \end{cases}$$

$$\begin{cases} J_3(q^{-1}) = J_2(q^{-1}) + F_2(0)q^{-2} = 1 + 2.5q^{-1} + 3.95q^{-2} \\ F_3(q^{-1}) = q\left[F_2(q^{-1}) - F_2(0).\overline{A}(q^{-1})\right] = 4.9250 - 7.0850q^{-1} + 3.16q^{-2} \end{cases}$$

Ainsi la matrice F est telle que :

$$F = \begin{bmatrix} 2.5 & -2.3 & 0.8 \\ 3.95 & -4.95 & 2.0 \\ 4.92 & -7.0850 & 3.16 \end{bmatrix}$$

On en déduit les éléments de G et H tels que :

$$\left|\begin{array}{l} B(q^{-1})J_1(q^{-1}) = G_1(q^{-1}) + q^{-1}H_1(q^{-1}) = \left(0.02 + 0.01q^{-1}\right) * (1) \\ \Rightarrow \begin{cases} G_1(q^{-1}) = 0.02 \\ H_1(q^{-1}) = 0.01 \end{cases} \\ \\ B(q^{-1})J_2(q^{-1}) = G_2(q^{-1}) + q^{-2}H_2(q^{-1}) = \left(0.02 + 0.01q^{-1}\right) * \left(1 + 2.5q^{-1}\right) \\ \Rightarrow \begin{cases} G_2(q^{-1}) = 0.02 + 0.06q^{-1} \\ H_2(q^{-1}) = 0.025q^{-2} \end{cases} \\ \\ B(q^{-1})J_3(q^{-1}) = G_3(q^{-1}) + q^{-3}H_3(q^{-1}) = \left(0.02 + 0.01q^{-1}\right) * \left(1 + 2.5q^{-1} + 3.95q^{-2}\right) \\ \Rightarrow \begin{cases} G_3(q^{-1}) = 0.02 + 0.06q^{-1} + 0.1040q^{-2} \\ H_3(q^{-1}) = 0.0395q^{-3} \end{cases} \end{array}\right.$$

La matrice H est donc :

$$H = \begin{pmatrix} 0.01 \\ 0.025 \\ 0.0395 \end{pmatrix}$$

D'âpres la formulation du Prédicteur optimal et la minimisation du la fonction de cout quadratique, la matrice G est selon les commande future (q croissant) et G est de dimension $(N_2 - N_1) \times N_u$, donc :

$$G = \begin{pmatrix} 0.02 & 0 \\ 0.06 & 0.02 \\ 0.1040 & 0.06 \end{pmatrix}$$

On obtient donc la matrice M

$$M = Q.G^T = \begin{pmatrix} 0.0932 & 0.2763 & 0.4745 \\ -0.0034 & 0.0880 & 0.2768 \end{pmatrix}$$

Avec Q tel que :

$$Q = \left[G^T.G + \lambda.I_{N_u}\right]^{-1} = \left[\begin{pmatrix} 0.02 & 0 \\ 0.06 & 0.02 \\ 0.1040 & 0.06 \end{pmatrix}^T \times \begin{pmatrix} 0.02 & 0 \\ 0.06 & 0.02 \\ 0.1040 & 0.06 \end{pmatrix} + 0.2\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\right]^{-1}$$

$$\Leftrightarrow Q = \begin{pmatrix} 4.6610 & -0.1700 \\ -0.1700 & 4.9082 \end{pmatrix}$$

Avec la stratégie de l'horizon fuyant, seule la première valeur de la séquence est appliquée au système.

D'ou :

$$m_1 = \begin{bmatrix} 0.0932 & 0.2763 & 0.4745 \end{bmatrix}$$

Finalement :

$$m_1 * F = \begin{pmatrix} 3.6614 & -4.9441 & 2.1267 \end{pmatrix}$$
$$m1 * H = 0.0266$$

La commande optimale trouvée est donc:

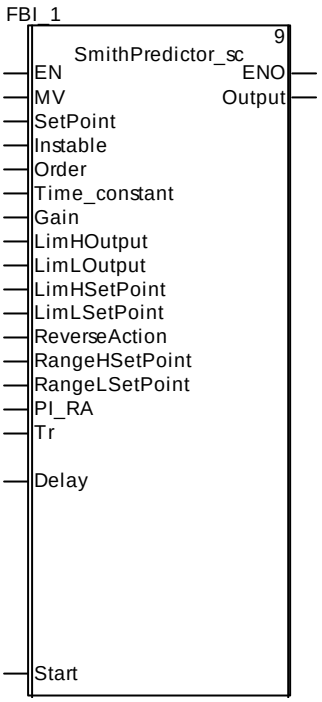
$$\Delta u_{opt}(t) = m_1{}^T . \left[w - if(q^{-1}).y(t) - ih(q^{-1}).\Delta u(t-1) \right]$$
$$\Leftrightarrow \Delta u_{opt}(t)\left[1 + 0.0266q^{-1}\right] = -\left[3.6614 - 4.9441q^{-1} + 2.1267q^{-2}\right]y(t)$$
$$+ \left[0.0932q + 0.2763q^2 + 0.4745q^3\right]w(t)$$

Annexe C – Algorithmes avancés de régulation dans l'automate Schneider (UnityV2.1) – objets DFB

SmithPredictor_sc

Description

Objet qui réalise le Prédicteur de Smith dans le cas de systèmes stables (1er ordre et 2nd ordre) et intégrateurs retardés.



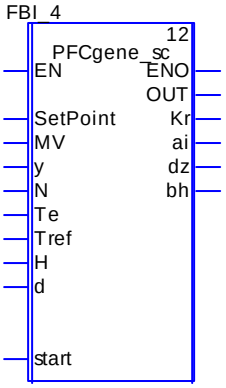
Entrées	Format	Description
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
MV	REAL	Valeur mesurée
SetPoint	REAL	Consigne
Instable	BOOL	0=systèmes stables, 1=systèmes intégrateurs
Order	INT	Coefficients du polynôme C [en2==coefficient de e(n-2)]
Time_constant	TIME	Constante de temps du système à corriger (systèmes stables)
Gain	REAL	Gain du système à corriger
LimHOutput	REAL	Limite haute de sortie contrôleur
LimLOutput	REAL	Limite basse de sortie contrôleur

LimHSetPoint	REAL	Limite haute de consigne contrôleur
LimLSetPoint	REAL	Limite haute de consigne contrôleur
ReverseAction	BOOL	Action inverse du contrôleur (1=actif)
RangeHSetPoint	REAL	Plage haute d'utilisation de la consigne
RangeLSetPoint	REAL	Plage basse d'utilisation de la consigne
PI_RA	BOOL	Action inverse relative au PI interne (systèmes stables)
Tr	REAL	Constante de temps désirée en boucle fermée (systèmes intégrateurs)
Delay	TIME	Retard du système identifié a corriger
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
Output	REAL	Sortie du contrôleur

PFCgene_sc

Description

Objet qui réalise la commande prédictive introduite par Richalet (PFC) dans le cas généralisée de systèmes auto-stables.



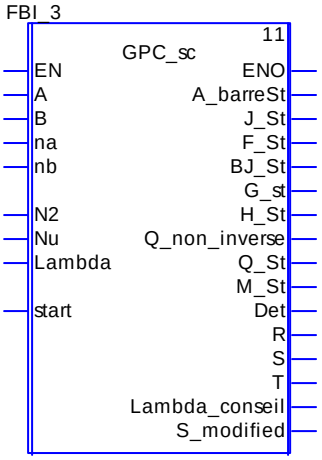
Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
SetPoint	REAL	Consigne
MV	REAL	Valeur mesurée
y	ARRAY[1…20] of REAL	Entrée permettant contenant le relevé de la sortie du système soumis a un échelon de mesure (nécessaire pour la mise en œuvre du régulateur) la sortie peut au plus contenir 20 mesures; cette fonctionnalité permet d'affranchir l'utilisateur des calculs des coefficients ai et dz.
N	INT	Nombre de mesure de y (doit être inferieur à 20)
Te	TIME	Entrée spécifiant l'échantillonnage du régulateur
TRef	TIME	Constante de temps désirée pour la trajectoire de référence
H	INT	Horizon pour le calcul de la commande
d	INT	Retard du système à corriger (en nombre d'échantillonnage) ; si Te=500ms et que le retard est de 2 secondes alors d=4.
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
OUT	REAL	Sortie du contrôleur
Kr	REAL	Sortie donnant le gain Kr du régulateur
ai	ARRAY[1…20]	Sortie contenant les coefficients ai (de la

	of REAL	représentation MA)
dz	ARRAY[1…20] of REAL	Sortie contenant les coefficients dz du régulateur prédictif
bh	REAL	Sortie contenant le bh du régulateur prédictif

GPC_sc

Description

Objet qui fait le calcul de la commande prédictive généralisées définie selon Clarke (GPC) et qui donne directement la forme RST du régulateur.



Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
A	ARRAY[1…10] of REAL	Polynôme A de la forme discrète du système à corriger
B	ARRAY[1…20] of REAL	Polynôme B de la forme discrète du système à corriger
na	INT	Degrés du polynôme A
nb	INT	Degrés du polynôme B
N2	TIME	Paramètre N2
Nu	TIME	Paramètre Nu
Lambda	INT	Paramètre λ
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
A_barreSt	ARRAY[1…11] of REAL	Coefficients du vecteur $\overline{A}$
J_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice J
F_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice F
BJ_St	ARRAY[1…11,1…11]	Coefficients de la matrice résultant du

	of REAL	produit B*J
G_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice G
H_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice H
Q_non_inverse	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice Q non inverse
Q_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice Q
M_St	ARRAY[1…11,1…11] of REAL	Coefficients de la matrice M
Det	REAL	Déterminant lors de l'inversion de Q
R	ARRAY[1…11] of REAL	Coefficients du polynôme R
S	ARRAY[1…11] of REAL	Coefficients du polynôme S (ce n'est pas le polynôme final à utiliser dans le correcteur RST)
T	ARRAY[1…11] of REAL	Coefficients du polynôme T
Lambda_conseil	REAL	Paramètre λ conseillé
S_modified	ARRAY[1…11] of REAL	Coefficients du polynôme S après introduction de l'intégrateur dans la boucle ; c'est ce polynôme S qui doit être utilisé dans le correcteur RST !

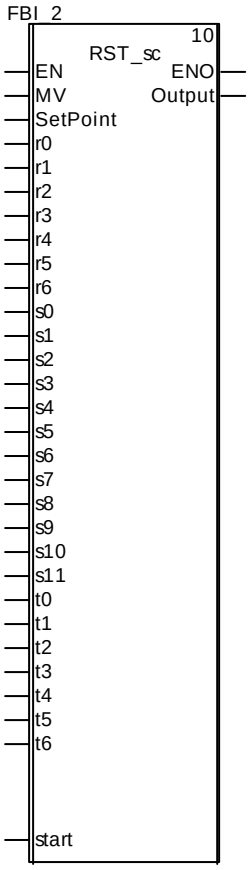
Remarque :

Cet objet possède des limitations inhérentes à l'implémentation sou UnityV3.2. Par exemple, le système ne doit être de degrés supérieur à 3 a cause du problème des déterminants dans le calcul des matrices inverses.

RST_sc

Description

Objet qui réalise le correcteur RST.

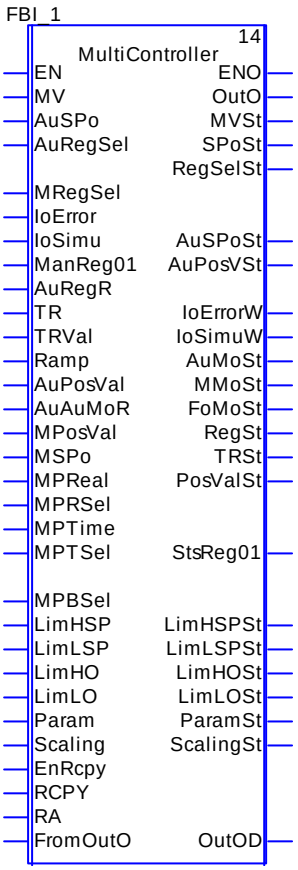


Entrées	**Format**	**Description**
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
r0 à r6	REAL	Coefficient du polynôme R
s0 à s11	REAL	Coefficient du polynôme S
t0 à t6	REAL	Coefficient du polynôme T
Start	BOOL	Démarre l'objet
Sorties	**Format**	**Description**
Output	REAL	Sortie du contrôleur

Remarque :

Cet objet possède des limitations inhérentes à l'implémentation sou UnityV3.2. Par exemple, le système ne doit être de degrés supérieur à 3 a cause du problème des déterminants dans le calcul des matrices inverses.

Annexe D – MultiController dans l'automate Schneider (UnityV2.1) – objets DFB



Entrées	Format	Description
EN	BOOL	Entrée toujours présente pour le synchronisme. Cette entrée doit toujours être connectée !
MV	REAL	Valeur mesurée
AuSPo	REAL	Consigne en mode automatique
AuRegSel	REAL	Sélection de la régulation en mode automatique (0=sélection du comportement 'Positioning')
MRegSel	REAL	Sélection de la régulation en mode manuel
IoError	BOOL	Pin spécifiant une erreur terrain sous jacente
IoSimu	BOOL	Pin spécifiant une simulation sous jacente
ManReg01	WORD	Registre des requêtes venant de l'utilisateur (HMI)
TR	BOOL	Tracking
TRVal	REAL	Position de sortie du contrôleur en Tracking
Ramp	REAL	Rampe de la consigne

AuPosVal	REAL	Position de sortie en mode automatique lors du comportement 'Positioning'
AuAuMoR	BOOL	Requête programme pour basculer l'objet en mode automatique
MPosVal	REAL	Position de sortie en modes manuels lors du comportement 'Positioning'
MSPo	REAL	Consigne en modes manuels
MPReal	REAL	Valeur d'un paramètre de type REAL de la structure Param qui est envoyé du HMI en modes manuels
MPRSel	REAL	Sélection du paramètre REAL de la structure Param à modifier lorsqu'une requête (en modes manuels) est faite (marche avec MPReal)
MPTime	REAL	Valeur d'un paramètre de type Time de la structure Param qui est envoyé du HMI en modes manuels
MPTSel	REAL	Sélection du paramètre TIME de la structure Param à modifier lorsqu'une requête (en modes manuels) est faite (marche avec MPTime)
MPBSel	REAL	Sélection du paramètre BOOL de la structure Param à modifier lorsqu'une requête (en modes manuels) est faite (marche avec le mot ManReg01)
LimHSP	REAL	Limite haute de la consigne
LimLSP	REAL	Limite basse de la consigne
LimHO	REAL	Limite haute de la sortie du contrôleur
LimLO	REAL	Limite basse de la sortie du contrôleur
Param	Structure PReal[1 to 21] PTime[1 to 5] PBool[1 to 5]	Structure générique des paramètres Organisation de la structure : [21 réels pour les paramètres de type REAL, 5 réels pour les paramètres de type TIME, 5 booléens pour les paramètres de type BOOL]
Scaling	Structure ScalOK SPRanMin SPRanMax PRanMin PRanMax	Structure dédiée au scaling
EnRcpy	BOOL	Validation de la fonction de recopie (RCPY)
RCPY	REAL	Recopie
RA	BOOL	Action inversée du régulateur si RA mis a 1

FromOutO	REAL	Copy de la sortie OutO pour la détermination de OutOD
Sorties	**Format**	**Description**
OutO	REAL	Sortie du contrôleur
MVSt	REAL	Valeur active de la valeur mesurée
SPoSt	REAL	Valeur active de la consigne appliquée au contrôleur
RegSelSt	REAL	Valeur active du type de régulation utilisée en comportement de régulation.
AuSPoSt	REAL	Valeur active de la consigne en mode automatique en comportement de régulation
AuPosVSt	REAL	Valeur active de la position du contrôleur en automatique lors du comportement en position (positioning)
IoErrorW	BOOL	Valeur active de l'erreur de terrain sous jacente
IoSimuW	BOOL	Valeur active de la simulation sous jacente
AuMoSt	BOOL	Pin d'indication sur le mode automatique (1 : mode auto actif)
MMoSt	BOOL	Pin d'indication sur le mode manuel (1 : mode manuel actif)
FoMoSt	BOOL	Pin d'indication sur le mode manuel forcée (1 : mode manuel forcée actif)
RegSt	BOOL	Pin d'indication sur le comportement du contrôleur (1 : régulation, 0 : position (positioning))
TRSt	BOOL	Pin d'indication si le Tracking est actif
PosValSt	REAL	Valeur active de la position de sortie du contrôleur en comportement de position (Positioning)
StsReg01	WORD	Registre de 'status' communiqué au HMI
LimHSPSt	REAL	Valeur active de la limite haute de la consigne
LimLSPSt	REAL	Valeur active de la limite basse de la consigne
LimHOSt	REAL	Valeur active de la limite haute de la sortie du contrôleur
LimLOSt	REAL	Valeur active de la limite basse de la sortie du contrôleur
ParamSt	Structure	Valeurs actives des paramètres de la structure Param
ScalingSt	Structure	Valeurs actives des paramètres de la structure Scaling
OutOD	REAL	OutOD=Out(n)-Out(n-1)

Annexe E – Méthodes d'identification et de modélisation graphiques

1. Etude graphique intuitive

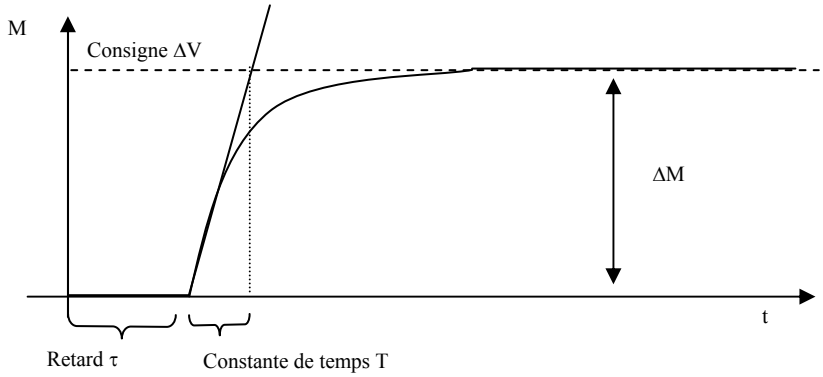
A partir de la réponse d'un procède soumis à un échelon, il est envisageable d'étudier intuitivement le système qu'il soit stable ou instable.

Paramètres procédés stables : assimilation a un premier ordre retardé

Dans le cas où le système est stable, le système peut être assimilé à un premier ordre retardé de la forme :

$$H(s) = \left. \frac{G.e^{-\tau s}}{1+Ts} \right|_{G=\Delta M / \Delta V}$$

L'intérêt majeur de cette approche empirique est qu'elle permet par un simple relevé d'obtenir une modélisation qui quelque fois est suffisante pour le réglage des contrôleurs.



Paramètres d'un procédé stable du premier ordre retardé

Le point d'inflexion n'est pas toujours simple à trouver. Dans ce cas les méthodes de Broida ou de Strejc peuvent répondre à ce problème.

Paramètres procédés instables : assimilation à un intégrateur retardé

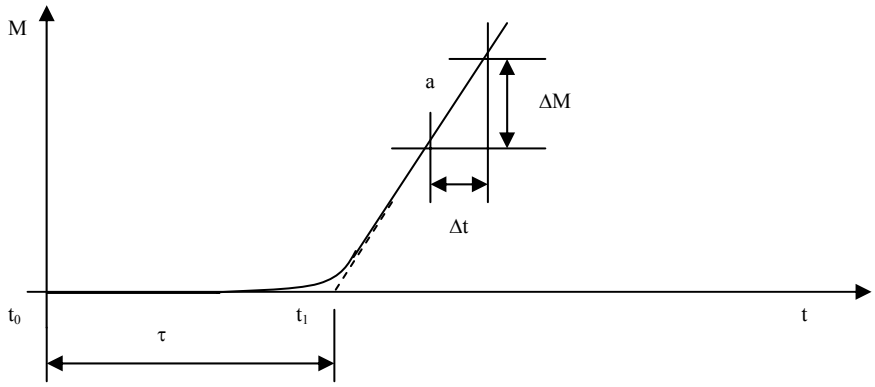
Dans le cas où le système est instable, le système peut être assimilé à un intégrateur retardé de la forme :

$$H(s) = \left. \frac{ke^{-\tau s}}{s} \right|_{k=a/\Delta V}$$

Cette modélisation est très efficace car elle approxime aussi les modèles des systèmes intégrateurs d'ordre supérieur à 1 de la forme :

$$H(s) = \frac{k}{s(1+Ts)^n}$$

Bien que le modèle précédent soit une approximation moins précise que le modèle ci-dessus, il assure des résultats très exploitables pour le calcul d'actions de régulateurs. Par exemple, la méthode de Ziegler Nichols pour le calcul du correcteur PID utilise efficacement ce modèle simplifié.

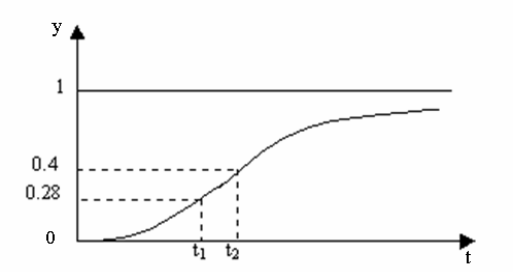


Paramètres d'un procédé instable du premier ordre retardé pour l'approximation d'un ordre n

2. Méthode de Broida

Le point d'inflexion étant mal défini, la méthode de Broida permet de trouver la constante de temps τ. et le temps mort R. Broida a proposé le calcul suivant pour approcher un système, à partir de sa réponse indicielle, par la fonction de transfert :

$$H(s) = \left.\frac{e^{-Rs}}{1+Ts}\right|_{\substack{R=2.8t_1-1.8t_2 \\ T=5.5(t_2-t_1)}}$$



Méthode graphique d'identification : la méthode de Broida

Il est également possible de calculer le retard et la constante de temps aux temps à 35.5% et 85.3% :

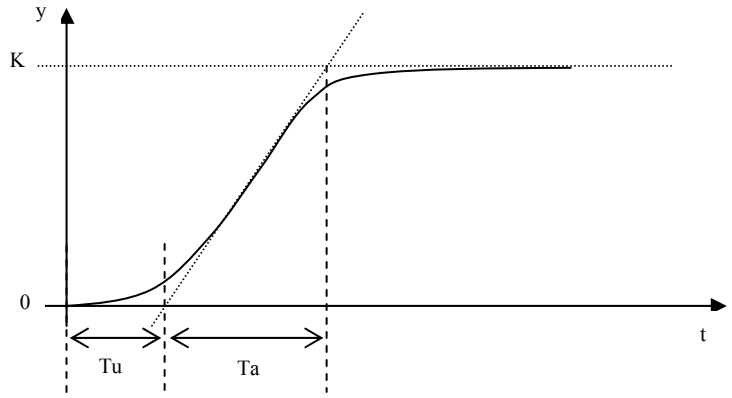
$$H(s) = \left.\frac{e^{-Rs}}{1+Ts}\right|_{\substack{R=1.3t_{35.3\%}-0.29t_{85.3\%} \\ T=0.67(t_{85.3\%}-t_{35.3\%})}}$$

3. Méthode de Strejc

Cette technique est basée sur les propriétés géométriques de la réponse indicielle. La méthode de Strejc consiste à identifier le système par un modèle d'ordre supérieur à 1 de la forme :

$$H(s) = \frac{G.e^{-\varpi}}{(1+Ts)^n}$$

L'idée est de tracer la tangente au point d'inflexion et de déterminer les valeurs Ta et Tu sur la courbe (voir courbe).



Méthode graphique d'identification : la méthode de Strejc

On calcul ensuite le rapport n=Tu/Ta et on cherche dans la table qui suit la valeur n immédiatement inférieur à celle calculée.

Tu/Ta	n	T/Ta
0	1	1
0.105	2	0.37
0.22	3	0.27
0.32	4	0.22
0.41	5	0.20
0.49	6	0.19
0.57	7	0.18
0.64	8	0.15
0.71	9	0.14
0.77	10	0.13

La constante de temps et le retard sont définis tels que :

$$T = {T}/{Ta}\Big|_{table}, \qquad \tau = Tu_{mesure} - Ta_{mesure}.{Tu}/{Ta}\Big|_{table}$$

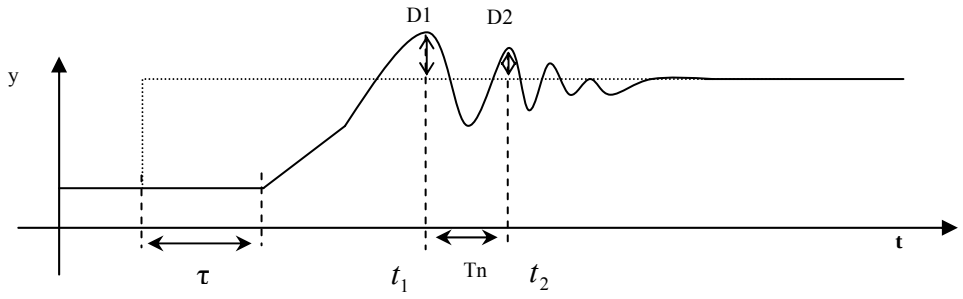
4. Méthode d'identification des systèmes oscillants

Cette dernière technique permet d'identifier les systèmes oscillants par un modèle du second ordre retardé :

$$H(s) = \frac{K.e^{-\varpi}}{1 + \frac{2\xi}{w_n}s + \frac{1}{w_n^2}s^2}$$

A partir des deux premiers dépassements D1 et D2, on détermine la pulsation et l'amortissement par les formules :

$$w_n = \frac{2\pi}{(t_2 - t_1)\sqrt{1-\xi^2}} \quad , \quad \frac{D_1}{D_2} = e^{\frac{2\pi\xi}{\sqrt{1-\xi^2}}}$$



Méthode graphique d'identification des systèmes oscillants

Bibliographie

Références bibliographiques

[Ale] G. Alevisakis and D.E. Seborg, An Extension of the Smith Predictor Method to Multivariable Linear Systems Containing Time Delays, Int. J. Control, vol. 18, 1973, pp. 541-551.

[Ast] K.J. Astrom, C.C. Hang, B.C. Lim, A New Smith Predictor for Controlling a Process with an Integrator and Long Dead-Time, IEEE Trans. Automatic, Vol.39, No.2, 1994.

[Bor] P. Borne, R.Ben Abdennour, M. Ksouri, F. M'sahli, Identification et commande numérique des procédés industriels, Technip, Paris, 2001.

[Bor93] P. Borne, G. Dauphin-Tanguy, J.-P. Richard, F. Rotella, I. Zambettakis, Analyse et régulation des processus industriels, Tome 2 – Régulation numérique, Ed. Technip, 1993.

[Bou] P. Boucher, D. Dumur, La Commande Prédictive, Méthodes et Pratiques de l'Ingénieur (P.Borne), Editions Technip, 1996.

[Cab1005] S.Cabaret, H.Coppier, A.Rachid, R.Barillère, LHC GCS process tuning: selection and use of PID and Smith Predictor for the regulations of the LHC experiments' gas systems, ICALEPCS, 10/2005, Geneva (Switzerland).

[Cab0207] S.Cabaret, H.Coppier, A.Rachid, R.Barillère, framework and Model Driven approaches for the 23 Gas Control Systems of the LHC experiments at CERN: practices for a SCADA-PLC industrial based solution, IMSM07, 02/2007, Buenos Ares (Argentina).

[Cab0807] S.Cabaret, H.Coppier, A.Rachid, R.Barillère, GPC and PFC with a Programmable Logic Controller, MMAR, 08/2007, Szczecin (Poland).

[Cab0907] S.Cabaret, From Identifying and Modeling to Control: the MultiController, an Advanced Control Loop Strategy for Cryogenic Systems at CERN, EUROSIM, 09/2007, Ljubljana (Slovenia).

[Cab1007] S.Cabaret, H.Coppier, A.Rachid, R.Barillère, Artem Burmyakov, MultiController: An Object Programming Approach to Introduce Advanced Control Algorithms for the GCS Large Scale Project, ICALEPCS, 10/2007, Knoxville (USA).

[Cla] David W. Clarke, Self-Tuning Control ("The Control Handbook" Section 53), Department of Engineering Science, Parks Road, Oxford, U.K.

[Cla87] D.W.Clarke, C. Mohtadi and P.S Tuffs, Generalized Predictive Control – Part I. The Basic Algorithm, Vol.23, No2, pp.137-148, Generalized Predictive Control – Part II. Extensions and Interpretations, Vol.23, No2, pp.149-160, Automatica 1987

[Cam] E.F Camacho, Constrained Generalized Predictive Control, IEEE Transactions on Automatic Control, Vol. 38, No.2., 01/02/93.

[Don] J. F. Donoghue. A comparison of the Smith Predictor and optimal design approaches for systems with delay in the control, IEEE Trans. on Ind. Elec. Cont. Instrum. vol. IECI-24. No 1, pp. 109-1 17. Feb 1977.

[Ede] Eder H., PID Tuning – science, art or both?, september 2005, Control Engineering Europe

[Eyk] P. Eykhoff, System Identification: Parameter and State Estimation, John Wiley, Londres, 1974

[Ful] A.T. Fuller, Optimal Nonlinear Control of Systems With Pure Delay, Int. J. Control, vol. 8, no. 2, 1968, pp 145-168.

[GCS03] R. Barillère et al., LHC GCS: A homogeneous approach for the control of the LHC experiment gas systems, ICALEPCS'2003, Gyeongiu, Korea, October 2003.

[GCSf05] G. Thomas et al., LHC GCS: A framework for the production of 23 homogeneous industry-like control systems, ICALEPCS'2005, Geneva, Switzerland, October 2005.

[GCSmd05] G. Thomas et al., LHC GCS: A Model-Driven approach for the automatic PLC and SCADA code generation, ICALEPCS'2005, Geneva, Switzerland, October 2005.

[Gra76] D. Graupe, Identification of Systems - 2nd edition, Robert E. Krieger Publishing Company, New York, 1976

[Gra03] Y. Granjon., AUTOMATIQUE, Systèmes linéaires, non linéaires, à temps continu, à temps discret, représentation d'état, Dunod, Paris, 2003

[Jac] Ivar Jacobson, Object-Oriented Software Engineering – A Use Case Driven Approach, 1992, ACM press, ADDISON-WESLEY.

[JCO] Oliver Holme, Manuel Gonzalez Berges, Piotr Golonka, Sascha Schmeling, The JCOP framework, ICALEPCS'2005, Geneva, Switzerland, October 2005.

[Jim] Thèse de F. F. Jimenez Fraustro, Conception sûre des automatismes industriels : modélisation synchrone de langages d'automates programmables de la norme CEI-61131-3, Université de Renne 1, 28/11/2001.

[JMF] Jean-Marie Flaus, La régulation industrielle, Hermes Science Publications, Paris, 1994

[Joh] Johnson R. E., frameworks = (Components + Patterns), Communication of the ACM, 40 (10), Oct. 1997, pp. 39-42

[Lan] I.D. Landau, Commande des systèmes – conception, identification et mise en œuvre, Hermès Science Publication, Paris, 2002.

[LET] Thèse de LE Anh Tuyet, Fédération : une Architecture Logicielle pour la Construction d'Applications Dirigée par les Modèles, 23/01/2004.

[Lju] L. Ljung, System Identification - Theory for the user, Prentice Hall PTR, 1987.

[Long06] Roland Longchamp, Commande Numérique de Systèmes Dynamiques – cours d'automatique, deuxième édition, Presses Polytechniques et universitaires romandes, 2006.

[Mat96] M.R. Matausek and A.D. Micic, A Modified Smith Predictor for Controlling a Process with an Integrator and Long Dead-Time, IEEE Transactions on Automatic Control, Vol. 41, no. 8, August 1996.

[Mat00] Thèse de Michael Mattsson, Evolution and Composition of Object-Oriented frameworks, University of Karskrona/Ronneby, 2000.

[OMG] Joaquim Miller and Jishnu Mukerji, OMG, Model Driven Architecture (MDA) - Document number ormsc/2001-07-01, July 9 2001.

[OMG03] Object Management Group Staff + Contributors, MDA Guide Version 1.0.1, 12/06/03.

[OMG01] Object Management Group Staff, A Proposal for an MDA Foundation Model - An ORMSC White Paper V00-02, 05/04/01.

[OMG00] Richard Soley and OMG Staff Strategy Group, Model Driven Architecture, 27/11/00.

[Pig] Pigeron B., Mullot H., Chaix A., Felix L., Aubert Y., Boucles de Régulation, 3eme édition 1996, Bhaly Autoédition.

[Ram] Gérard Ramond, Contribution à la Commande Prédictive Généralisée Adaptative et Applications, thèse UniversitéParis XI, UFR Scientifique d'Orsay, 28/09/2001.

[Ric04] J. Richalet, G. Dreyfus, G. Lavielle, J. Mallet, La commande prédictive, Mise en œuvre et applications industrielles, Eyrolles, 2004.

[Ric93] J. Richalet, Pratique de la commande prédictive, ADERSA, Hermes, 1993.

[Smi] Smith O.J.M., Atherton D.P., A controller to overcome dead band time, ISA J., vol. 6, no. 2 pp.28-33, 1959

[UNI] P. Gayet et al., UNICOS a framework to built industry-like control systems, Principles and Methodology, ICALEPCS'2005, Geneva, Switzerland, October 2005.

[Vei] J. Veille Application de la norme ISA88 pour la spécification fonctionnelle dans les projets d'automatismes, Instrumentation & Systèmes - Octobre 2003.

[Wat] K. Watanabe. A process-Model Control for Linear Systems with Delay, IEEE Trans. Automatic Control, vol.AC-26,No6, p1261-1266, 1981.

[Zha] Zhuang M., Atherton D.P., Automatic tuning of optimum PID controllers, IEE Proceedings-D, Vol.140, No.3, May 1993

[Zig] Ziegler J. G.; Nichols N. B., Optimum settings for automatic controllers, 1942, Trans. ASME

Sources Internet

[LHC] The LHC GCS project, http://itcofe.web.cern.ch/itcofe/Projects/LHC-GCS/

[Wik] 'Wikipédia- l'encyclopédie libre', http://fr.wikipedia.org/wiki/Framework

Résumé

Cette thèse présente les résultats obtenus dans le cadre de mes recherches au CERN (Centre Européen pour le Recherche Nucléaire). Elle traite de l'intégration d'algorithmes de contrôles avancés pour les 21 installations à Gaz du nouvel accélérateur de particule, le LHC (Large Hadron Collider). La démarche s'inscrit dans le choix stratégique du projet appelé GCS (Gaz Control System) : produire l'ensemble des systèmes de contrôle commande par des déclinaisons récursives de production automatique des applications. Les équipements de contrôles prévus à cet effet sont des automates programmables industriels et des systèmes de supervision industriels. Dans ce contexte, certains algorithmes de contrôles avancés se démarquent par leur capabilité à s'adapter à ces exigences : le prédicteur de Smith (compensation des retards), les méthodes prédictives (robustesses et réglages conviviaux), le contrôleur RST (structure ouverte et résolution polynomiale) et les identifications paramétriques en ligne (utilisation des modèles de comportement). Les travaux s'articulent en quatre points. Le premier traite de la synthèse des stratégies d'identifications et de régulations avancées dans un automate programmable industriel. Le deuxième exprime la validité d'implémentation. Le troisième développe la systématisation d'utilisation d'algorithmes de régulation, dans un projet empreint de deux méthodologies de réalisation innovantes : un framework de développement et une réalisation dirigée par les modèles. Enfin le quatrième s'attache à fournir aux experts de processus à gaz, une interface standard permettant l'utilisation efficace des algorithmes de régulation avancés.