UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN SCIENCES DES PÂTES ET PAPIERS

PAR
MOHAMED CHBEL

CONCEPTION D'UN MODULE JACOBIEN EN SIMULATION ET SON APPLICATION EN CONTRÔLE AVANCÉ

JUIN 2007

Avant-propos

Le contrôle des procédés est de plus en plus important dans l'industrie papetière qui traverse des moments difficiles pour rentabiliser ses produits. La non-linéarité du procédé entraîne des grands changements au niveau du gain du procédé ce qui affecte la stabilité et la performance du système de contrôle. En contrôle multivariable, une variable manipulée peut affecter plusieurs variables contrôlées, ce qui provoque des interactions des boucles. Le design d'un système de contrôle efficace passe donc par le choix de gains optimaux selon la non-linéarité du procédé (point d'opération) ainsi que par le choix des paires de variables contrôlées-manipulées qui minimisent l'interaction à partir de la matrice des gains relatif.

Le projet de maîtrise a pour objectif la conception d'un module Jacobien dans le logiciel de simulation commercial Cadsim Plus, qui permet le calcul des gains en boucle ouverte et le calcul de la Matrice des Gains Relatifs. Ceci permet des applications au niveau du contrôle avancé telles que la programmation des gains et le contrôle multivariable.

Remerciements

Je voudrais exprimer mes sincères remerciements à mon directeur, M. Luc Laperrière, pour son aide et ses conseils d'expert en simulation et contrôle des procédés qui étaient très pertinent tout le long du projet.

Finalement, je remercie ma conjointe et ma famille pour leur encouragement.

Résumé

Les procédés chimiques en général ont un comportement non-linéaire où les propriétés dynamiques changent en fonction du point d'opération. L'ajustement des paramètres du contrôleur passe par une meilleure connaissance de la dynamique du procédé.

En contrôle multivariable, le choix des paires de variables contrôlées-manipulées doit prendre en compte l'interaction des boucles de contrôle, la Matrice des Gains Relatifs (MGR) offre un outil efficace pour minimiser les interactions.

Les outils de simulation sont de plus en plus utilisés pour le design et l'optimisation du contrôle, le logiciel Cadsim Plus en est un exemple, il permet aux utilisateurs de créer leur propre module de simulation appelés « Dynamic Load Modules » ou DLM.

L'objectif de la recherche est la conception d'un module de simulation appelé Jacobien, qui calcule les gains du procédé en boucle ouverte pour ajuster le gain du contrôleur à un point d'opération donné, en plus de calculer la MGR pour un meilleur choix des paires de variables du système de contrôle.

Trois études de cas ont été utilisées pour montrer l'utilité d'un tel module pour améliorer la performance du contrôle.

Dans le cas du concentrateur de liqueur noire, la non-linéarité affecte la stabilité du contrôleur, le module Jacobien a réussi à réajuster le gain du contrôleur au point d'opération choisi par l'utilisateur.

Dans la simulation de lavage de pâte chimique, la programmation des gains qui utilise les gains optimaux à partir du module Jacobien, offre un meilleur contrôle des solides dissous en diminuant autant la variabilité que les coûts d'opérations par rapport au contrôle ordinaire et manuel.

Finalement, dans le contrôle multivariable d'un mélangeur simple, le module Jacobien a permis le calcul de la matrice des gains relatif qui offre à l'utilisateur un meilleur choix

des paires de variables contrôlée et manipulée. Ce choix peut être différent selon la plage d'opération.

En résumé le module Jacobien développé est un outil très intéressant pour améliorer la performance du contrôleur dans le logiciel de simulations Cadsim Plus.

Juin 2007

Mots clés

Contrôle de procédés, contrôleur PID, contrôle avancé, matrice des gains relatifs, contrôle multivariable, Cadsim Plus, optimisation , interaction des boucles, gains en boucle ouverte, programmation des gains, procédés non-linéaires.

Table des matières

Liste des figures

Liste des tableaux

Liste des équations

$$K'_{21} = \frac{K_{21}}{\lambda_{21}} = \frac{jacobien(dy_2 / dx_1)}{RGA(dy_2 / dx_1)} = \frac{-0{,}286}{0{,}86} = -0{,}33\ (\% / kg/s)$$ Éq. 6.1 83

$$y_1 = x_1 + x_2$$ Éq. 6.2 86

$$y_1 * y_2 = x_1 * c_1 + x_2 * c_2$$ Éq. 6.3 86

$$y_2 = \frac{x_1 * c_1 + x_2 * c_2}{x_1 + x_2}$$ Éq. 6.4 86

$$K_{ij} = \lim_{\Delta x_j \to 0} \frac{\Delta y_i}{\Delta x_j} = \frac{\partial y_i}{\partial x_j}$$ Éq. 6.5 86

$$K_{11} = \frac{\partial y_1}{\partial x_1} = 1$$ Éq. 6.6 86

$$K_{12} = \frac{\partial y_1}{\partial x_2} = 1$$ Éq. 6.7 86

$$K_{21} = \frac{\partial y_2}{\partial x_1} = \frac{-x_2(c_2 - c_1)}{(x_1 + x_2)^2}$$ Éq. 6.8 86

$$K_{22} = \frac{\partial y_2}{\partial x_2} = \frac{x_1(c_2 - c_1)}{(x_1 + x_2)^2}$$ Éq. 6.9 86

$$\lambda_{ij} = \begin{matrix} \\ y_1 \\ y_2 \end{matrix} \begin{pmatrix} \overset{X1}{\dfrac{K_{11}\,K_{22}}{K_{11}K_{22} - K_{12}K_{21}}} & \overset{X2}{\dfrac{-K_{12}\,K_{21}}{K_{11}K_{22} - K_{12}K_{21}}} \\ \dfrac{-K_{12}\,K_{21}}{K_{11}K_{22} - K_{12}K_{21}} & \dfrac{K_{11}\,K_{22}}{K_{11}K_{22} - K_{12}K_{21}} \end{pmatrix}$$ Éq. 6.10..... 87

$$\lambda_{ij} = \begin{matrix} \\ y_1 \\ y_2 \end{matrix} \begin{pmatrix} \overset{X1}{\lambda_{11} = \dfrac{x_1}{x_1 + x_2}} & \overset{X2}{\lambda_{12} = \dfrac{x_2}{x_1 + x_2}} \\ \lambda_{21} = \dfrac{x_2}{x_1 + x_2} & \lambda_{22} = \dfrac{x_1}{x_1 + x_2} \end{pmatrix}$$ Éq. 6.11..... 87

Liste des abréviations

EDO	Équations différentielles ordinaires
DLM	Dynamic Load Module
FD	Facteur de dilution
MGR	Matrice des gains relatifs (en anglais RGA : Relative gain array)
MIMO	Multiple inputs, multiple outputs
PID	Contrôleur proportionnel intégral dérivatif
RD	Rapport de déplacement
SISO	Single input single output
T	Température
t	Temps
x	Variable manipulée ou indépendante
y	Variable contrôlée ou dépendante

Chapitre 1 - Introduction

1.1 Importance du contrôle des procédés

L'être humain a toujours cherché à contrôler son environnement pour subvenir à ses besoins de survie, de bien être et de développement. Ce comportement qui passe parfois inaperçu est utilisé chaque jour, comme pour contrôler la température de notre douche ou maintenir notre vitesse de voiture selon les conditions ambiantes.

Pour l'industrie, le contrôle des procédés est d'une importance majeure, car pour maintenir une bonne qualité des produits tout en préservant la sécurité des employés et la protection de l'environnement dans une économie de plus en plus compétitive, le système de contrôle doit être performant. Dans ce système le contrôleur représente le cerveau qui va agir au moyen d'algorithmes mathématiques sur le procédé pour atteindre les objectifs souhaités.

L'industrie papetière est d'autant plus concernée par le contrôle des procédés vue les réglementations environnementales de plus en plus sévères et les difficultés qu'affronte cette industrie pour rentabiliser ses produits. Les procédés papetiers requièrent plusieurs étapes de transformation chimiques et mécaniques avec l'apport de beaucoup d'énergie surtout pour les pâtes mécaniques de raffineurs où le contrôle de l'énergie spécifique a un grand impact sur les propriétés des fibres et la consommation d'énergie du raffinage.

Le besoin du contrôle des procédés peut se manifester dans les trois objectifs suivants [1]:

Sécurité : Les procédés industriels comportent souvent des risques potentiels pour la sécurité des employés ainsi que pour la population. La manipulation des substances dangereuses et l'utilisation des équipements, doivent satisfaire à des conditions bien précises de température et de pression par exemple, pour éviter des bris du matériel et les dangers d'explosions ou d'incendies. Le même risque pour la sécurité peut survenir pour des produits chimiques inoffensifs au départ, mais qui deviennent dangereux une

fois mélangés à d'autres produits, ou manipulés à des pressions ou températures prohibées par les fournisseurs.

Réglementations environnementales : Les lois fédérales et provinciales limitent de plus en plus les émissions à l'environnement pour préserver la qualité de l'air et de l'eau car ils ont un impact direct sur la santé des êtres vivants [2]. Le contrôle des procédés papetier a pour objectif de réduire les émissions des produits chimiques toxiques à l'environnement à un niveau acceptable fixé par les normes en vigueurs par un traitement efficace des effluents en plus de réduire la consommation en eau grâce aux circuits fermés.

Considérations économiques : Le profit représente le but ultime, la raison d'être, de toute entreprise à but lucratif. Pour générer des profits, il faut minimiser les coûts de production tout en gardant une bonne qualité du produit final. Dans l'industrie papetière, le papier doit répondre à des spécifications de résistance à la traction et du poids de base bien précis pour son utilisation dans les imprimeries. Une qualité médiocre fera certainement fuir les clients, mais en même temps, une qualité supérieure aux spécifications du client entraînera un coût de production élevé et une baisse du profit. Le système de contrôle a pour objectif de garder la qualité dans la fourchette spécifiée, entre la valeur maximale et minimale.

1.2 Revue historique

Le principe de contrôle automatique par rétroaction était déjà utilisé par les Grecs en 250 A.J. pour contrôler le niveau d'eau par l'intermédiaire d'un flotteur. Cependant le régulateur à boules de James Watt en 1788 est considéré comme le début du principe d'automatisation au niveau industriel, il permettait la régulation de la vitesse d'une machine à vapeur [3].

Les années 1930 représentent une avancée non négligeable en contrôle des procédés, où en 1936 on a commercialisé le premier contrôleur PID pour application industrielle. Le contrôle pneumatique a continué de dominer jusqu'à l'apparition des contrôleurs électroniques à la fin des années 50. Le début des années 60 a connu le développement

des premiers systèmes de contrôle par ordinateur avec des possibilités intéressantes de contrôle à des coûts acceptables, entre autres, le contrôle cascade, ratio, gain adaptatif et contrôle par anticipation [1].

Le développement de nouvelles techniques tel que le contrôle flou, le contrôle prédictif et les réseaux neuronaux ont beaucoup contribué à des avancements au niveau du contrôle des procédés [4].

1.3 Modélisation dynamique des procédés

L'ajustement des paramètres du contrôleur est essentiel pour le bon fonctionnement du procédé et la qualité du contrôle. Pour optimiser ces paramètres, il faut comprendre les caractéristiques dynamiques du procédé à contrôler en s'appuyant sur des modèles mathématiques qui renferment des informations précieuses sur la prédiction du comportement du procédé face aux changements des conditions d'opération, en plus d'identifier les variables qui devraient être contrôlées ou manipulées dans la stratégie de contrôle [5].

La modélisation mathématique d'un procédé chimique peut se faire selon deux approches différentes. Le modèle théorique s'appuie sur les lois de la conservation de la masse et de l'énergie en plus des lois physiques et chimiques. Cette méthode est avantageuse car elle représente un modèle réaliste et valide pour toute la plage d'opération possible du procédé en question. Cependant, le grand désavantage de cette méthode est la difficulté d'obtenir des solutions précises aux équations différentielles générées à partir des procédés complexes et non-linéaires qui nécessitent le passage par la linéarisation.

La deuxième approche consiste à utiliser les données du procédé réel pour générer un modèle empirique ou « modèle de boite noire » qui décrit la relation qui lie les variables d'entrées à celles de la sortie. C'est une méthode facile à développer, mais elle comporte un désavantage majeur d'extrapolation du modèle lors d'un changement des conditions d'opération. Bien que les paramètres du modèle n'aient pas toujours une signification physique comparativement à ceux du procédé, comme les coefficients de transfert de

chaleur ou de masse, cette méthode de modélisation est efficace et moins coûteuse en temps de développement que la modélisation théorique [6].

On peut aussi développer un modèle semi-empirique, en combinant les deux approches mentionnées. Ce type de modèle serait valable pour une plage d'opération plus large avec moins d'efforts de développement théorique.

Quoi qu'il en soit, en modélisant un procédé, il faut comprendre que le modèle n'est jamais unique, et qu'il est juste une approximation du procédé réel.

Les modèles dynamiques ont des avantages considérables dans le contrôle des procédés [6] :

a) **La compréhension du procédé :** Les modèles dynamiques ainsi que les simulations, permettent de comprendre le comportement du procédé, en plus d'extraire des informations précieuses sur sa dynamique sans perturbation et en sauvant temps et argent.

b) **La formation des opérateurs :** Les simulateurs de procédé peuvent jouer un rôle clé dans la formation du personnel pour opérer des unités complexes et réagir rapidement et efficacement en cas de conditions d'urgence.

c) **Le développement des stratégies de contrôle :** De nouvelles stratégies de contrôle peuvent être mise en place en se basant sur le modèle dynamique pour identifier les variables du procédé qui doivent être contrôlées ou manipulées.

d) **L'optimisation des conditions d'opération :** Les conditions optimales d'opération sont importantes pour maximiser le profit tout en minimisant les coûts. Le modèle du procédé, dynamique ou en régime permanent, combiné à des informations économiques (coût en matières premières, énergie, etc.) permettent d'identifier le point d'opération le plus profitable.

1.4 Réponse dynamique des procédés

La résolution des équations qui modélisent le procédé permet de décrire le comportement dans le temps des variables de sortie par rapport à un changement dans les variables d'entrées. Il existe plusieurs méthodes pour résoudre des systèmes d'équations différentielles comme l'intégration analytique ou numérique surtout pour des systèmes non-linéaires complexes. Pour des équations différentielles ordinaires linéaires (EDO) où la dérivée est par rapport à une seule variable indépendante (le temps), la transformée de Laplace est un outil mathématique de résolution puissant très utilisé en contrôle des procédés. Elle permet de générer la fonction de transfert. Cette dernière est une expression algébrique de la relation dynamique entre une variable d'entrée et une autre de sortie. Elle est indépendante des conditions initiales et sa forme standard permet à l'utilisateur de reconnaître assez facilement le comportement du procédé en question [6].

Chaque procédé possède ses propres caractéristiques dynamiques qui le différentient des autres. En générale, les procédés papetiers peuvent être caractérisés par des modèles de premier ordre avec ou sans délai (débit, consistance, pression), de deuxième ordre (température) ou intégrateurs (niveau) [1].

1.5 Le contrôle par rétroaction

Le principe de fonctionnement d'un contrôle par rétroaction est simple et très utilisé, il suffit de mesurer la variable à contrôler et la comparer avec la valeur désirée (point de consigne) et ensuite agir sur la variable à manipuler. On utilise souvent ce type de contrôle dans notre vie quotidienne, comme pour ajuster la température de l'eau en ouvrant ou fermant le robinet de l'eau chaude ou froide ou encore ajuster notre vitesse de voiture en appuyant sur l'accélérateur ou les freins selon les conditions ambiantes (vitesse du vent, pente de la route).

En général un système de contrôle est fondé sur trois opérations de base [7] :

1. **Mesure :** Mesurer la variable à contrôler par un senseur et (ou) un transmetteur.

2. **Décision :** En se basant sur la mesure, le contrôleur décide, au moyen d'algorithmes et de logique, de la façon à agir pour maintenir la variable contrôlée à la valeur désirée.

3. **Action :** Le système doit agir sur la variable à manipuler en réponse à la décision du contrôleur.

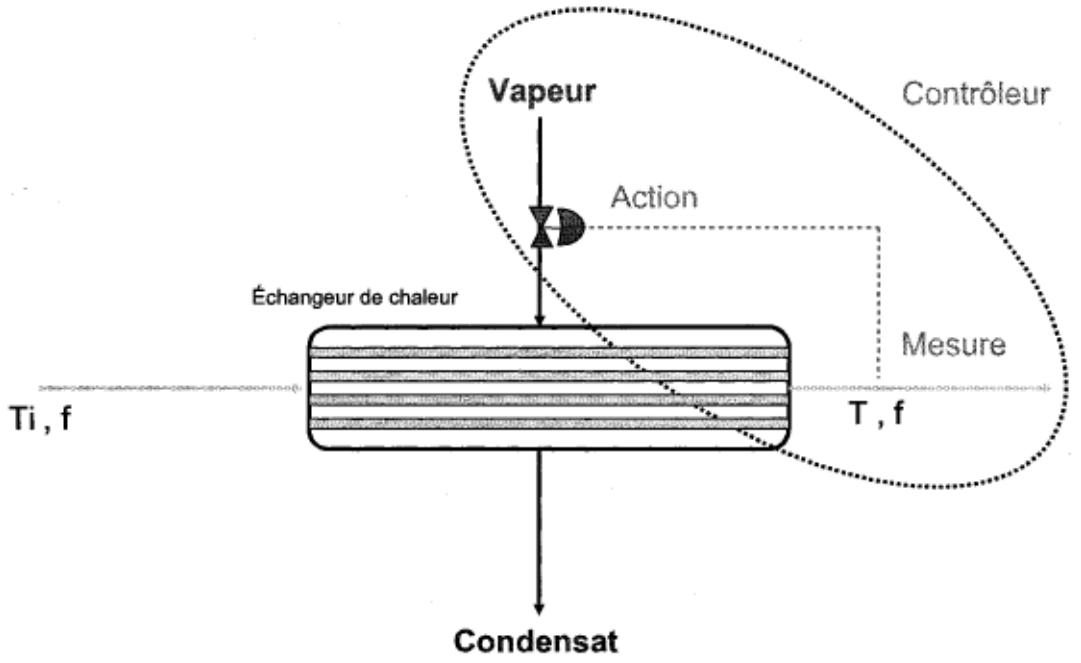


Figure 1.1 Système de contrôle d'un échangeur de chaleur.

La figure 1.1 représente un système de contrôle par rétroaction de la température d'un liquide à la sortie de l'échangeur de chaleur avec une variable contrôlée et une variable manipulée (SISO). T(t) est la variable contrôlée tandis que la variable manipulée est l'ouverture (0-100%) de la valve de vapeur qui contrôle le débit. En comparant la valeur actuelle de la température T(t) avec le point de consigne, le contrôleur doit agir sur la valve pour ramener la température de sortie à la valeur désirée.

Plusieurs autres facteurs peuvent faire dévier la température T(t) de son point de consigne, ce sont les perturbations. Dans l'exemple de la figure 1.1 les perturbations du

système de contrôle sont dûes aux fluctuations de la température d'entrée T_i (t) et du débit f(t), des conditions ambiantes et de la composition du fluide. Souvent ces perturbations ne sont pas mesurées, ce qui représente un désavantage du système de contrôle par rétroaction, où le contrôleur ne commence à réagir qu'après que la variable contrôlée dévie du point de consigne. Le contrôle par anticipation a l'avantage de réagir rapidement aux perturbations lorsqu'elles surviennent, sauf que ces dernières doivent être mesurées et modélisées, ce qui est très désavantageux. Une combinaison des deux stratégies peut s'avérer très efficace où le contrôle par anticipation fait la compensation des perturbations majeures et le contrôle par rétroaction se charge d'amener la variable mesurée au point de consigne en réagissant aux autres perturbations non mesurées [7].

1.6 Ajustement des paramètres du contrôleur PID

L'algorithme de contrôle PID est le plus utilisé dans l'industrie pour le contrôle en rétroaction depuis plus de 50 ans. Son succès est dû à sa flexibilité et sa puissance. Il est composé de trois modes de contrôle de base, à savoir, les modes de contrôle proportionnel, intégral et dérivatif (Annexe 1). La majorité des contrôleurs utilisé dans l'industrie papetière (99 % et plus) utilise un algorithme de type PID [1]. Cet algorithme calcule la valeur de la sortie du contrôleur à utiliser pour agir sur la variable à manipuler (ouverture de valve ou vitesse de la pompe) de telle sorte que l'écart entre la mesure de la variable à contrôler et son point de consigne soit diminué à zéro idéalement.

L'ajustement des paramètres d'un contrôleur PID est essentiel pour que la boucle de contrôle satisfait aux critères de performance suivants [6]:

- Stabilité du système en boucle fermée.

- Effet des perturbations minimisé.

- Réponse rapide et moins oscillatoire après un changement de point de consigne.

- Élimination l'erreur en régime permanent «Offset».

- Le système de contrôle doit être robuste, ce qui implique les paramètres du contrôleur PID permettent un contrôle acceptable pour une large plage d'opération.

En réalité, on ne peut pas satisfaire toutes ces conditions simultanément car la performance et la robustesse sont deux objectifs conflictuels. En effet, la performance a pour but d'augmenter la rapidité de la réponse et de minimiser les oscillations, tandis qu'on peut atteindre la robustesse en choisissant une stratégie de contrôle plus conservatrice et moins performante avec un gain K_c plus faible et une plus grande valeur de τ_I la constante du temps intégrale.

Plusieurs méthodes d'ajustement des paramètres du contrôleur (Annexe 1) se basent sur un modèle du procédé à contrôler comme la méthode de synthèse directe, le modèle de contrôle interne et la simulation sur ordinateur. Pour l'ajustement en ligne, la méthode de Ziegler-Nichols est la plus connue, néanmoins elle est moins populaire auprès des techniciens et ingénieurs de contrôle des procédés en pâtes et papiers car elle génère beaucoup d'oscillations et d'instabilité. Une des méthodes la plus recommandée en pâtes et papier est la méthode Lambda qui offre un contrôle avec une grande stabilité [8].

Chapitre 2 - Problématique

2.1 Problématique d'ajustement des modèles non-linéaires

Les procédés chimiques représentent en général un comportement non-linéaire très difficile à analyser et à résoudre où les propriétés dynamiques changent [9]. La linéarisation autour d'un point d'opération nous permet d'utiliser la transformée de Laplace et ainsi trouver la fonction de transfert reliant la variable de sortie à celle de l'entrée. Cette linéarisation est une approximation du comportement non-linéaire et elle est seulement valide autour du point utilisé comme base à la linéarisation (point d'opération) [10]. Pour effectuer la linéarisation d'un modèle non-linéaire, l'expansion en séries de Taylor est la méthode la plus répondue.

2.1.1 Linéarisation avec les séries de Taylor

Un système est linéaire obéit au principe de superposition où les propriétés d'additivité et de proportionnalité sont satisfaites :

$$F(ax_1 + bx_2) = aF(x_1) + bF(x_2) \qquad \text{Éq. 2.1}$$

Lorsque le modèle est non-linéaire, il faut faire une approximation linéaire de chaque terme autour d'un point d'opération en utilisant les séries de Taylor :

Fonction avec une seule variable

$$f(x) = f(x_s) + \left.\frac{df}{dx}\right|_{x_s} (x - x_s) + \frac{1}{2!}\left.\frac{d^2 f}{dx^2}\right|_{x_s} (x - x_s)^2 + R \qquad \text{Éq. 2.2}$$

Fonction à deux variables

$$F(x_1, x_2) = F(x_{1s}, x_{2s}) + \left.\frac{\partial F}{\partial x_1}\right|_{x_{1s}, x_{2s}} (x_1 - x_{1s}) + \left.\frac{\partial F}{\partial x_2}\right|_{x_{1s}, x_{2s}} (x_2 - x_{2s}) + \frac{1}{2!}\left.\frac{\partial^2 F}{\partial x_1^2}\right|_{x_{1s}, x_{2s}} (x_1 - x_{1s})^2$$
$$+ \frac{1}{2!}\left.\frac{\partial^2 F}{\partial x_2^2}\right|_{x_{1s}, x_{2s}} (x_2 - x_{2s})^2 + \frac{1}{2!}\left.\frac{\partial^2 F}{\partial x_1 \partial x_2}\right|_{x_{1s}, x_{2s}} (x_1 - x_{1s})(x_2 - x_{2s}) + R$$

Éq.2.3

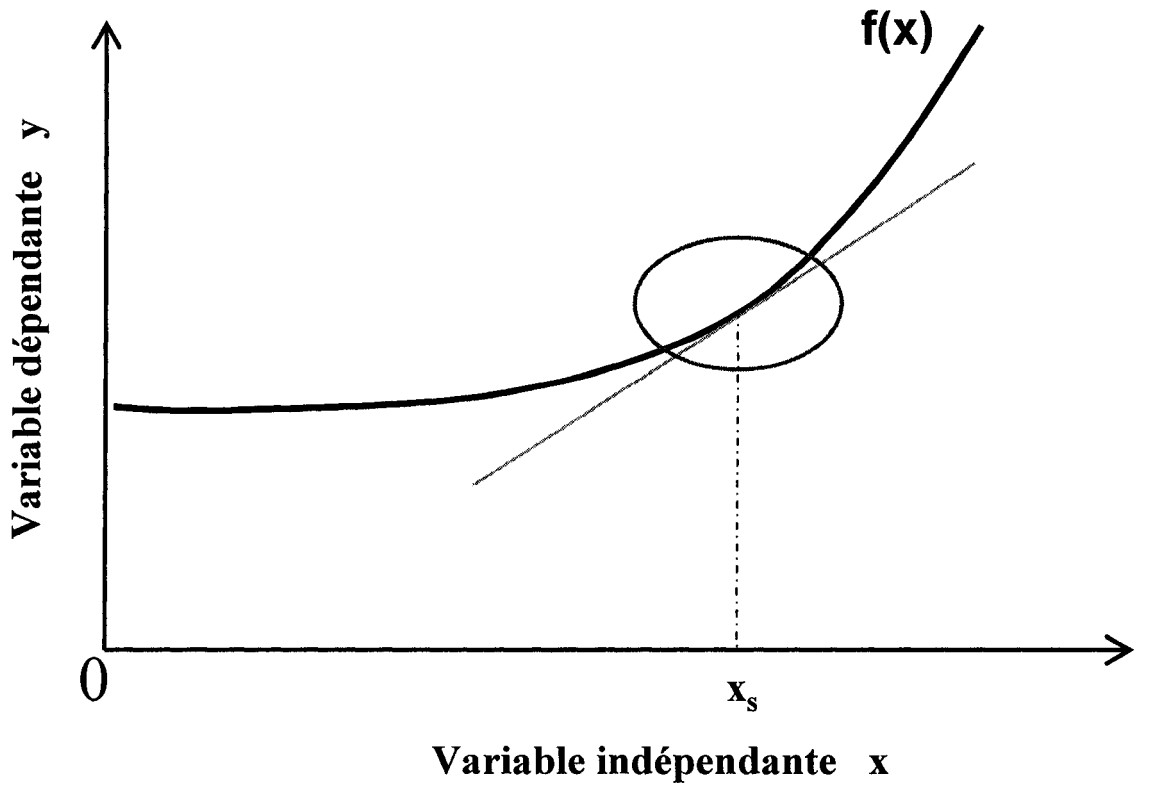


Figure 2.1 Linéarisation d'une fonction à une seule variable

La figure 2.1 représente une linéarisation d'une fonction f(x) autour d'un point de base x_s. On peut constater que la pente au point (x_s , $f(x_s)$) est une bonne approximation de la fonction dans la région délimité par le cercle. Cependant, à l'extérieur de cette zone le modèle linéaire est inapproprié.

En pratique, les procédés chimiques peuvent avoir un comportement non-linéaire ou légèrement non-linéaire (voir figure 2.2). On peut estimer que les caractéristiques du procédé telles que le gain et la constante de temps ne changent pas beaucoup en fonction du point d'opération lorsqu'on a affaire avec un procédé légèrement non-linéaire. Notre modèle de procédé est donc valide sur la plage d'opération définie lors du design et la performance du contrôleur ajustée à partir des paramètres du procédé est optimal. Dans le cas contraire, où on constate un changement significatif des paramètres du procédé, on est alors en présence d'un comportement non-linéaire.

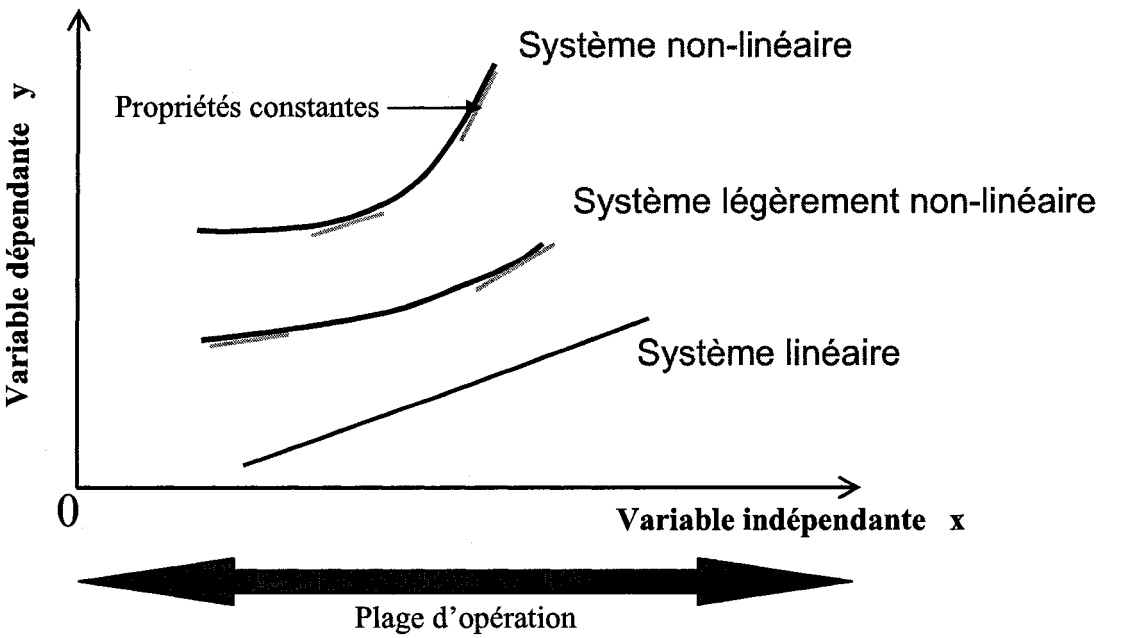


Figure 2.2 **Degrés de non-linéarité d'une fonction à une seule variable**

2.1.2 Influence de la non-linéarité sur la qualité de contrôle

L'effet de la non-linéarité sur le contrôle d'un procédé se fait sentir lors du changement des conditions d'opérations. Les variations au niveau du gain et de la constante du temps ont une influence sur la stabilité et la performance de la boucle de contrôle [11]. Les paramètres du contrôleur, calculés par différentes méthodes d'ajustement, sont estimés être optimaux autour du point d'opération ainsi que pour toute la plage d'opération pour des procédés ayant un comportement linéaire car les paramètres du procédé sont indépendants des conditions d'opération. En réalité les procédés en général comportent des non-linéarités plus au moins importantes. Les contrôleurs sont alors ajustés de telle façon à ce qu'on a une performance optimale autour du point d'opération du design et acceptable sur la plage d'opération possible du procédé [7].

Dans le cas d'un comportement non-linéaire sévère où les caractéristiques du procédé changent drastiquement à travers la plage d'opération, une combinaison fixe des paramètres du contrôleur ne suffit pas à garder une assez bonne performance, en plus

d'un risque élevé lié à la non stabilité de la boucle de contrôle. Le contrôle adaptatif peut alors s'avérer efficace avec des techniques comme la programmation des gains «gain scheduling», ou encore le contrôleur auto ajustable «self tuning controller».

L'impact d'un contrôleur mal ajusté sur le procédé peut se manifester en :

- Qualité médiocre du produit
- Gaspillage au niveau des produits utilisés et de l'énergie
- Haute variabilité et baisse de profit
- Risque sur la sécurité des employés et de l'environnement.

2.1.3 Programmation des gains

La méthode de programmation des gain est une technique de contrôle adaptatif assez simple et efficace, utilisée pour le contrôle des procédés non-linéaires. Le but est d'ajuster un ou plusieurs paramètres du contrôleur PID en se basant sur une variable mesurée appelée «scheduling variable» [12]. Cette dernière peut être la variable contrôlée, la variable manipulée ou autre variable mesurée du procédé. On choisit généralement une variable qui change lentement comme la variable contrôlée, au lieu de la variable manipulée qui change généralement plus rapidement [6].

Pour implanter la programmation des gains dans un contrôleur, on peut utiliser une des stratégies suivantes pour ajuster les paramètres du contrôleur selon le changement de la variable programmée :

- Les paramètres du contrôleur changent en continu avec la variable programmée [13]
- La variable programmée est divisée en régions dans lesquelles les caractéristiques du procédé sont peu différentes et donc on assume que le gain et la constante de temps sont constantes (figure 2.3). Pour chaque région on a une combinaison différente des paramètres du contrôleur. A noter qu'en général,

seulement le gain du contrôleur est réajusté car plusieurs procédés présentent des variations au niveau du gain mais une dynamique (constante de temps) assez constante [6].

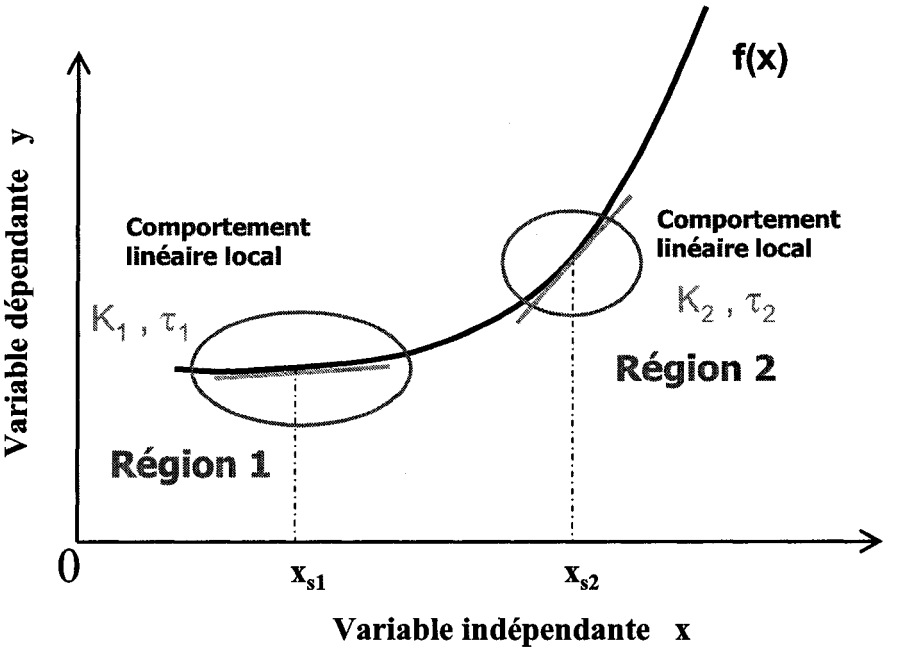


Figure 2.3 **Comportement linéaire local selon les régions de la plage d'opération**

Lorsque tous les paramètres du contrôleur sont trouvés pour chaque région de la plage d'opération, on peut alors les stocker dans un tableau et les utiliser, tout dépendant où on se trouve sur la plage d'opération, pour configurer le contrôleur (figure 2.4) [14].

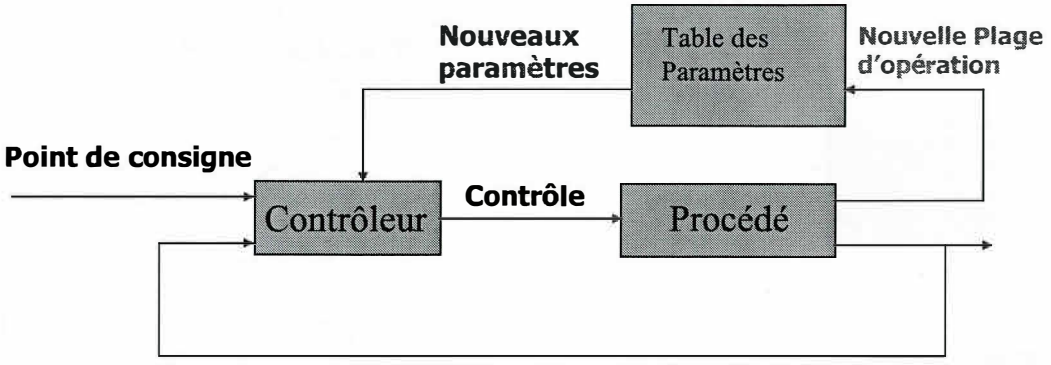


Figure 2.4 Diagramme bloc d'un système utilisant la programmation des gains

2.1.4 Contrôleur auto ajustable

Cette technique est surtout utilisée pour des procédés où la dynamique du procédé est imprédictible à cause des changements fréquents des perturbations non mesurables (qualité et débits de la matière première) [15].

La figure 2.5 montre le principe de fonctionnement d'un contrôleur auto-ajustable, où les paramètres du contrôleur sont calculés en continu suivant les changements de la dynamique du procédé sans intervention humaine. Ce type de contrôle reste néanmoins peu utilisé en industrie en le comparant avec la programmation des gains à cause des risques d'instabilité et d'insécurité des opérations. Il reste donc du chemin à faire pour améliorer les algorithmes utilisés [6].

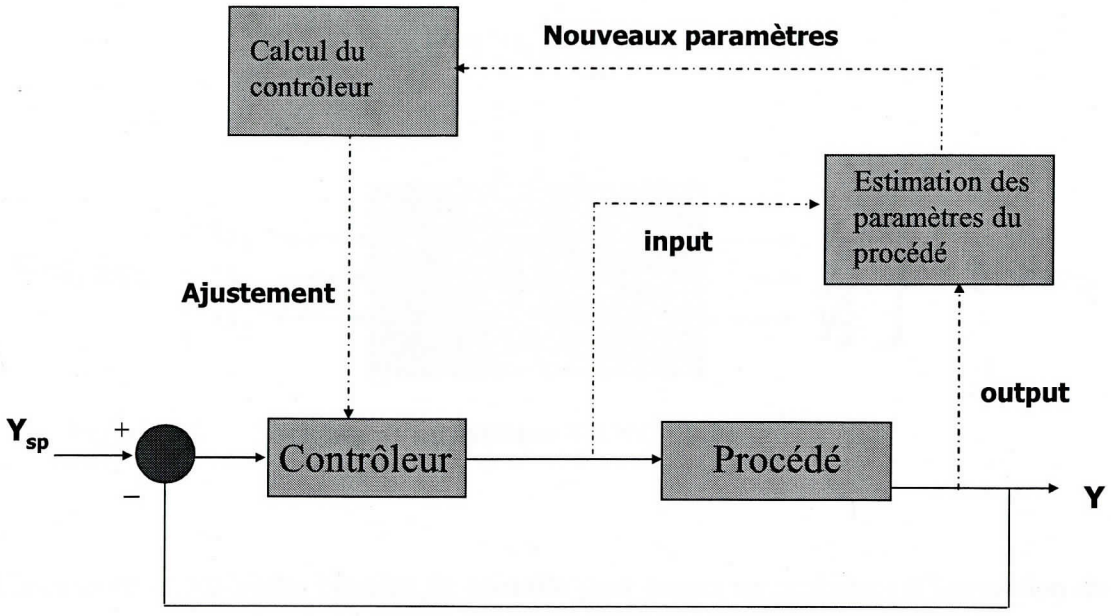


Figure 2.5 Diagramme bloc d'un système de contrôle auto ajustable.

L'usine Blandin aux É.U a utilisé un contrôleur adaptatif qui a permis de contrôler efficacement le blanchiment de la pâte [15]. Auparavant, c'est un opérateur qui contrôlait manuellement l'injection du peroxyde. Le contrôleur adaptatif a permis une diminution de la consommation des produits chimiques utilisés et une réduction de la variabilité de la blancheur. Ce mode de contrôle a permis à l'usine de produire à plusieurs spécifications d'indice de blancheur tout en réduisant ses coûts de production.

2.2 Problématique d'ajustement des modèles multivariable

Les systèmes étudiés dans les sections précédentes, se caractérisent par une seule variable à contrôler et une autre à manipuler. Dans ce système on analyse une seule boucle de contrôle.

En pratique, l'objectif de contrôle est de maintenir plusieurs variables à contrôler à leur point de consigne en manipulant une ou plusieurs variables [16]. Dans ce cas on a présence d'un système MIMO (figure 2.6) [3].

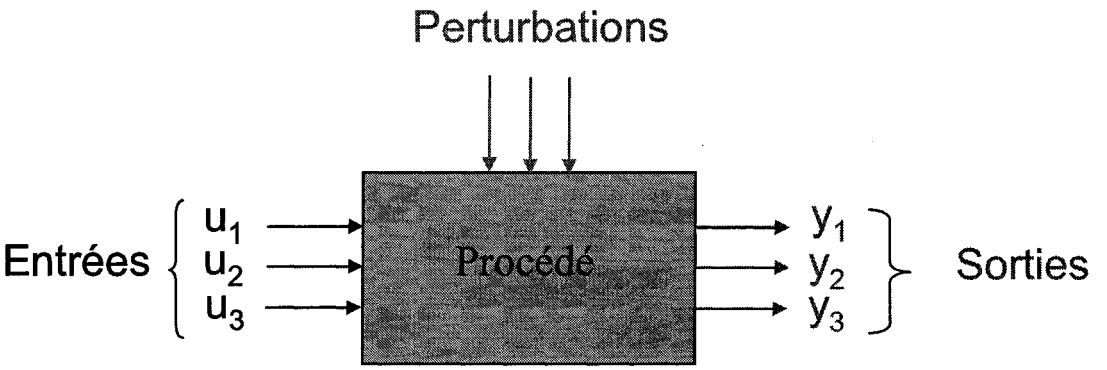


Figure 2.6 Schéma d'un système MIMO 3×3.

L'existence de multiples boucles de contrôle peut causer un problème d'interaction de boucles où la variable à manipulée affecte plus qu'une variable contrôlée [17]. La sélection de la meilleure paire de variables manipulée-contrôlée devient une tâche difficile avec l'augmentation du nombre des variables, et peut avoir une influence sur la performance et la stabilité du contrôle [18].

2.2.1 L'effet d'interaction des boucles

Un exemple simple de contrôle multivariable est illustré par la figure 2.7. Il s'agit d'un mélangeur avec deux flux à l'entrée : dilué et concentré. L'objectif du contrôle est de manipuler l'ouverture des valves pour atteindre le débit « w » et la concentration « x » souhaités à la sortie. Les deux paires de variables contrôlée-manipulée sont : «w-m_1» et «x-m_2», on verra plus loin que ce choix influence la qualité et la performance du contrôle.

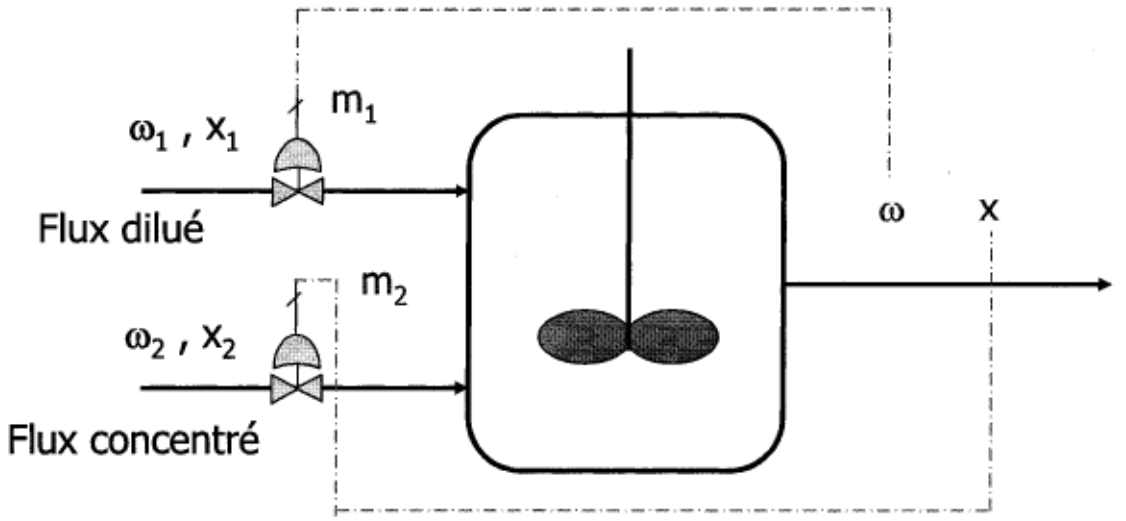


Figure 2.7 Contrôle multivariable d'un mélangeur (2×2).

La figure 2.8 montre la réponse du procédé lors d'un changement au niveau des variables manipulées «w_1» et «w_2». Au point A, les deux boucles de contrôles sont en mode manuel, on effectue alors un changement au débit concentré «w_2» tout en gardant le débit dilué «w_1» constant. On constate alors l'effet direct sur la concentration à la sortie «x» avec un gain positif dû à l'augmentation du débit concentré «w_2». Lorsqu'au point B on met la boucle de contrôle du débit diluée en mode automatique tout en gardant le débit concentré «w_2» constant, on s'aperçoit que la concentration «x» a augmenté, ce qui est normal car le contrôleur du débit a diminué le débit dilué «w_1» pour revenir au point de consigne. La différence entre l'effet total et l'effet direct sur la variable contrôlée «x» est due à l'effet d'interaction des deux boucles. Dans cet exemple, on a une «interaction positive» puisque l'effet d'interaction va dans la même direction du changement initial. Dans le cas contraire, on parle d'une «interaction négative» et les deux boucles se battent [7].

L'interaction des boucles peut avoir des conséquences néfastes pour le contrôle [19]:

- Instabilité des boucles et baisse des performance

- Augmenter le degré de difficulté au niveau de l'ajustement des paramètres du contrôleur

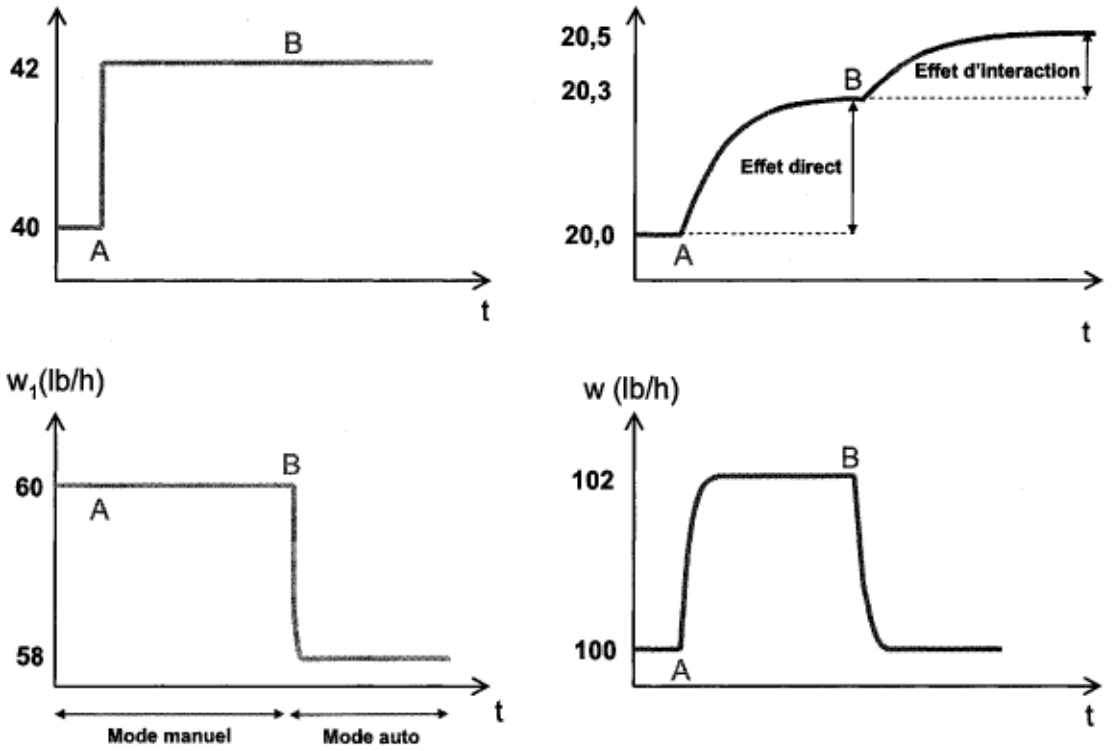


Figure 2.8 Effet des interactions sur les variables contrôlées x et w.

Pour diminuer l'effet d'interaction des boucles, il est nécessaire de bien choisir la paire de variables contrôlée-manipulée. Il serait un peu simpliste de choisir la paire en se basant sur les gains en boucle ouverte, qui démontrent quelle variable manipulée a plus d'influence sur la variable contrôlée. En effet les gains ont des unités différentes et ne peuvent donc pas être comparés même si on peut les exprimer en terme de pourcentage du signal du transmetteur par rapport à celui du contrôleur.

En 1966, Bristol a publié une méthode quantitative qui permet de déterminer la meilleure paire de variables contrôlée-manipulée dans un système multi-variable pour atténuer l'effet des interactions [20]. Cette technique permet de calculer la MGR qui mesure l'interaction des boucles de façon indépendante des unités d'ingénierie et des plages des transmetteurs ou des valves.

2.2.2 Calcul de la matrice MGR et son interprétation

Pour obtenir la MGR de Bristol, on a besoin de calculer le paramètre« λ_{ij} » tel que :

$$\lambda_{ij} = \frac{\text{gain entre la variable manipulée}(m_j)\text{et contrôlée}(c_i)\text{avec les boucles ouvertes}}{\text{gain entre la variable manipulée}(m_j)\text{et contrôlée}(c_i)\text{avec les boucles fermées}}$$

Éq. 2.4

Skogestad a démontré que les éléments de la MGR peuvent être calculés directement de la matrice des gains du procédé K en boucle ouverte, en effectuant le calcul matriciel suivant [21] :

$$MGR = K \otimes (K^{-1})^T \qquad \text{Éq. 2.5}$$

- Gains en boucle ouverte :

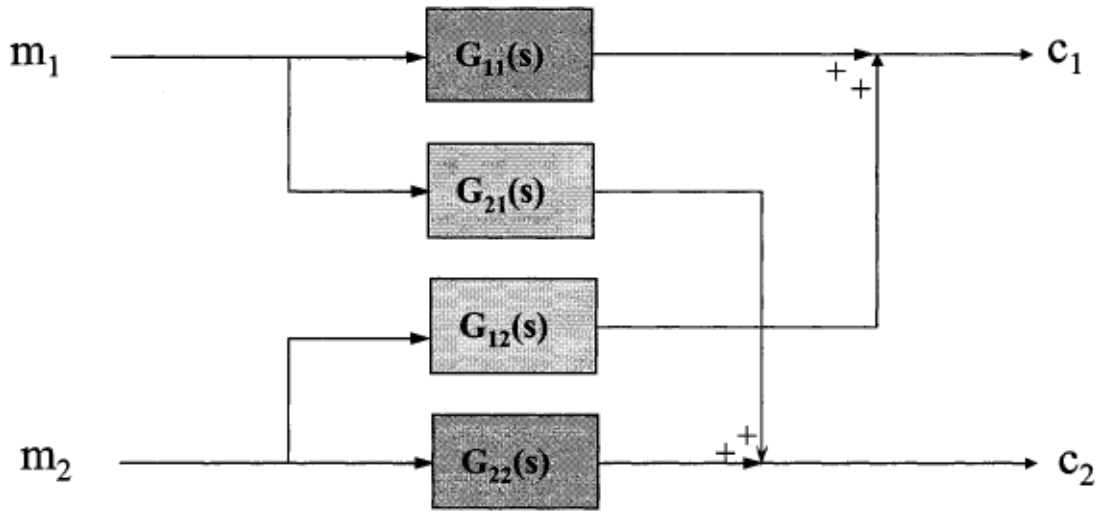


Figure 2.9 Diagramme bloc en boucle ouverte d'un système (2×2).

À partir de la figure 2.9, on peut déterminer l'effet de chaque variable manipulée sur les variables contrôlées en calculant les gains des fonctions de transfert pour un système

(2×2). Par exemple, les fonctions de transfert G_{11} et G_{21} représentent l'effet de la variable manipulée «m_1»sur les deux variables contrôlées «c_1» et «c_2» [21,22].

Les gains en boucle ouverte calculés lors d'un changement appliqué à m_1 tout en gardant m_2 constante sont :

$$K_{11} = \frac{\Delta c_1}{\Delta m_1} \qquad \text{Éq. 2.6}$$

$$K_{21} = \frac{\Delta c_2}{\Delta m_1} \qquad \text{Éq. 2.7}$$

La même procédure peut être appliquée à m_2 en gardant m_1 constante :

$$K_{12} = \frac{\Delta c_1}{\Delta m_2} \qquad \text{Éq. 2.8}$$

$$K_{22} = \frac{\Delta c_2}{\Delta m_2} \qquad \text{Éq. 2.9}$$

La matrice des gains en boucle ouverte d'un système (2×2) devient :

$$K = \begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix} \qquad \text{Éq. 2.10}$$

On peut calculer les éléments de la matrice MGR pour un système (n×n), en effectuant un calcul matriciel qui utilise seulement les gains en boucle ouverte :

$$\Lambda = MGR = K \otimes (K^{-1})^T = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1n} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ \lambda_{n1} & \lambda_{n2} & \lambda_{n3} & \lambda_{nn} \end{bmatrix} \qquad \text{Éq. 2.11}$$

Notons que la somme des colonnes ou des lignes équivaut à 1.

Maintenant que la matrice MGR est calculée, il faut interpréter les valeurs des gains relatifs λ_{ij} [21,22]:

1. $\lambda_{ij} = 1$, il n'y a pas d'interaction avec les autres boucles de contrôle.

2. $\lambda_{ij} = 0$, la variable manipulée «j», n'a pas d'effet sur la variable contrôlée «i».

3. $\lambda_{ij} = 0.5$, haut degré d'interaction. Les autres boucles ont le même effet sur la variable contrôlée «i» que la variable manipulée j.

4. $0.5 < \lambda_{ij} < 1$, l'interaction existe, néanmoins il faut garder la paire choisie pour minimiser l'effet de l'interaction.

5. $\lambda_{ij} > 1$, l'interaction réduit le gain effectif de la boucle de contrôle. Des valeurs supérieures à 10, indiquent que le système est sensible aux petites variations du gain.

6. $\lambda_{ij} < 0$, le gain en boucle fermée est à l'opposé du gain en boucle ouverte. À éviter.

En suivant les indications ci-dessus, on peut donc recommander d'éviter de faire des paires pour des gains relatifs négatifs ou nuls, et privilégier le cas où ces gains relatifs sont proches de l'unité [23].

L'analyse des interactions avec la matrice des gains relatif a été très utile dans un procédé de raffineurs de pâte thermomécanique, l'article [24] montre les interactions qui affectent la qualité de la pâte et la stratégie de contrôle décentralisé qui prend en compte les interactions des boucles à partir de la matrice des gains relatif.

2.3 Les objectifs de la recherche

Le survol de la littérature scientifique concernant le contrôle des procédés et particulièrement le contrôleur PID, a permis de faire ressortir deux problématiques importantes qui ont un impact direct sur la performance et la stabilité du contrôle. La problématique de contrôle des procédés non-linéaires requiert un réajustement des paramètres PID lors d'un changement de niveau des conditions d'opération. Dans la plupart des cas, un réajustement du gain du contrôleur permet d'atteindre la performance souhaitée et éviter les problèmes d'instabilité [22]. Avec la méthode Lambda, qui est très utilisé dans le domaine des pâtes et papier [1], on a besoin d'évaluer le gain de procédé pour réajuster le gain proportionnel du contrôleur PID.

Une autre problématique tout aussi importante, à savoir le contrôle multi-variable, requiert de l'ingénieur de choisir la meilleure paire de variables manipulée-contrôlée pour minimiser les effets d'interaction. Il existe pour cela une technique qui calcule la MGR en se basant sur les gains en boucle ouverte entre les variables contrôlées et manipulées.

L'objectif de cette recherche est de concevoir et programmer un module dans le logiciel de simulation Cadsim Plus qui sera capable d'évaluer les gains entre des variables dépendantes ou contrôlées et celles indépendantes ou manipulées et calculer la MGR de façon automatique et fiable.

Le module conçu est appelé «Jacobien» puisqu'il évalue des dérivées partielles «dy/dx», il pourrait être très utile dans une simulation sur Cadsim Plus pour faire les opérations suivantes :

- Ajuster le gain du contrôleur lors d'un changement des conditions d'opération
- Évaluer le gain à différents points d'opération pour détecter la non-linéarité

- Générer la MGR qui offre à l'utilisateur la possibilité de choisir la meilleure paire de variables contrôlée-manipulée pour minimiser l'effet de l'interaction des boucles.

- Le module peut calculer les dérivées d'une fonction à un point donné ce qui est très utile pour trouver des minimums et maximums d'une fonction, donc pour une utilisation potentielle en optimisation (méthodes par gradients) ou en solution de systèmes d'équations non-linéaires (Méthode de Newton-Raphson).

Chapitre 3 - Méthodologie

3.1 Introduction

Pour trouver les paramètres optimaux d'un contrôleur, la simulation représente un avantage majeur car elle permet d'optimiser le procédé de façon sécuritaire, en économisant temps et argent. Cadsim Plus est un logiciel de simulation de procédé de type séquentiel modulaire, dans lequel le procédé est formé d'un agencement de modules où on fournit des variables d'entrée de chaque module et l'on calcule les variables de sortie de façon séquentielle selon les bilans massique et énergétique.

Dans ce simulateur, les paramètres du contrôleur PID doivent être estimés par l'utilisateur en se basant sur la dynamique du procédé à contrôler. Sachant que la grande majorité des procédés ont un comportement non-linéaire, l'estimation des paramètres optimaux du contrôleur PID est nécessaire pour chaque nouveau point d'opération où les propriétés dynamiques du procédé (gain et constante de temps) changent.

Pour améliorer cette lacune, l'objet de cette recherche est la conception, programmation et validation d'un module qui calcule le Jacobien (matrice de dérivées partielles) et donc les gains K_p, est d'une grande utilité pour estimer les gains des contrôleurs K_c pour bonne une performance et la stabilité de la boucle de contrôle.

La matrice des gains en boucle ouverte entre les variables contrôlées et manipulées calculée par le module Jacobien, est aussi utilisée pour le calcul de la MGR qui permet de faire une analyse pour un contrôle multivariable. Le module Jacobien pourrait aussi servir de base à certains algorithmes d'optimisation du procédé puisqu'il calcule des dérivées partielles.

3.2 Conception du module Jacobien

3.2.1 Description du module

Le module Jacobien développé permet à l'utilisateur de calculer les gains (dérivées partielles) entre les variables indépendantes «x_j» (manipulées) et les variables dépendantes «y_i» (contrôlées) et ensuite générer la MGR.

En mode dessin de Cadsim Plus, l'utilisateur doit dessiner un polygone et le nommer Jacobien ou « Jacobian » pour qu'il soit reconnu par le logiciel. Ensuite, il faut spécifier 2 chaînes de caractères, à savoir «NumIndependant = nombre de variables indépendantes» et «NumDependant = nombre de variables dépendantes». Maintenant l'utilisateur peut passer au mode spécification et suivi de la simulation pour définir les paramètres du module et les variables dépendante et indépendante du module Jacobien.

3.2.2 Paramètres du module

Le module Jacobien contient trois paramètres qui doivent être spécifiés par l'utilisateur:

- « Bump » (0 ou 1)
- « Settling time »
- « Bump factor »

a) «Bump»

Le paramètre «Bump» sert comme un interrupteur, initialement sa valeur est à 0, la simulation continue sans calcul du module Jacobien. Lorsque l'utilisateur veut déclencher un calcul du module Jacobien, il doit mettre la valeur du «Bump» à 1.

b) « Settling time »

Le temps de stabilisation (en anglais « Settling time ») correspond au temps nécessaire pour que la variable contrôlée «y_i» (variable dépendante) atteigne sa valeur finale en

régime permanent, après une excitation de la variable manipulée «x_j» (variable indépendante). Chaque procédé a un temps de stabilisation différent, il incombe à l'utilisateur de définir ce temps en effectuant un ««Bump»» manuel de la variable manipulée «x_j» et déduire le temps nécessaire qui permettra la stabilisation de «y_i». L'utilisateur peut aussi faire une perturbation du module Jacobien avec un temps de stabilisation assez large et ensuite déduire à partir du graphique de «y_i» le temps de stabilisation adéquat.

Le module Jacobien peut calculer des gains entre plusieurs variables «y_i» et «x_j». Il faut donc mettre un temps de stabilisation assez grand pour permettre à toutes les variables dépendantes «y_i» d'atteindre leur valeur en régime permanent. L'utilisateur est conseillé de faire une première perturbation d'essai, pour ensuite mettre le bon temps de stabilisation.

Un temps de stabilisation trop court fausse inévitablement le calcul du gain qui va se répercuter aussi sur le calcul du gain du contrôleur et la MGR.

Il faut faire attention aux procédés intégrateurs comme le contrôle de niveau d'un réservoir, la variable dépendante «y_i» ne se stabilise pas après une perturbation de la variable indépendante «x_j». Il faut choisir un temps de stabilisation court pour éviter un débordement ou un assèchement du réservoir.

c) «Bump factor »

Le facteur de perturbation (en anglais «Bump factor ») représente l'amplitude de l'excitation de la variable indépendante x_j selon l'équation suivante :

$$x_j(excitée) = x_j(initiale) * (1 + Bump\ factor) \qquad \text{Éq. 3.1}$$

La variable x_j perturbée, correspond à la variable calculée « CurrentIndependantVar » du module Jacobien (annexe 2).

Le facteur de perturbation est une valeur absolue qui peut être positif ou négatif. Plus sa valeur est proche de 0, plus la précision du gain augmente pour atteindre la dérivée de y_i par rapport à x_j de la façon suivante :

$$K_{ij} = \lim_{\Delta x_j \to 0} \frac{\Delta y_i}{\Delta x_j} = \frac{\partial y_i}{\partial x_j} \qquad \text{Éq. 3.2}$$

Le facteur de perturbation ne doit pas être nul à cause de la division par zéro.

Kij représente le gain entre la variable indépendante x_j et dépendante y_i.

En contrôle des procédés, l'ajustement des paramètres du contrôleur se fait par perturbation en mode manuel de 5% à 15% de la sortie du contrôleur ce qui correspond à un facteur de perturbation de 0,05 à 0,15.

L'utilisateur doit choisir un facteur de perturbation adéquat pour le type de procédé.

3.2.3 Lien des variables du module avec les variables du procédé

Pour effectuer les calculs des gains, il faut définir les variables indépendantes et dépendantes par ordre numérique. On présente ici un exemple de simulation avec 3 variables indépendantes «x_1, x_2, x_3» et 3 variables dépendantes «y_1, y_2, y_3».

Après création du module Jacobien et spécification des nombres de variables indépendantes et dépendante et les paramètres du module, une série de questions est posée par le logiciel en mode spécification [25].

Le module Jacobien commence par questionner l'utilisateur sur la valeur correspondante à la variable indépendante1. L'utilisateur doit choisir la spécification locale et le type d'équation «égaler à une autre variable» puis placer l'icône sur la conduite correspondante à la variable «x_1» comme le montre la figure 3.1.

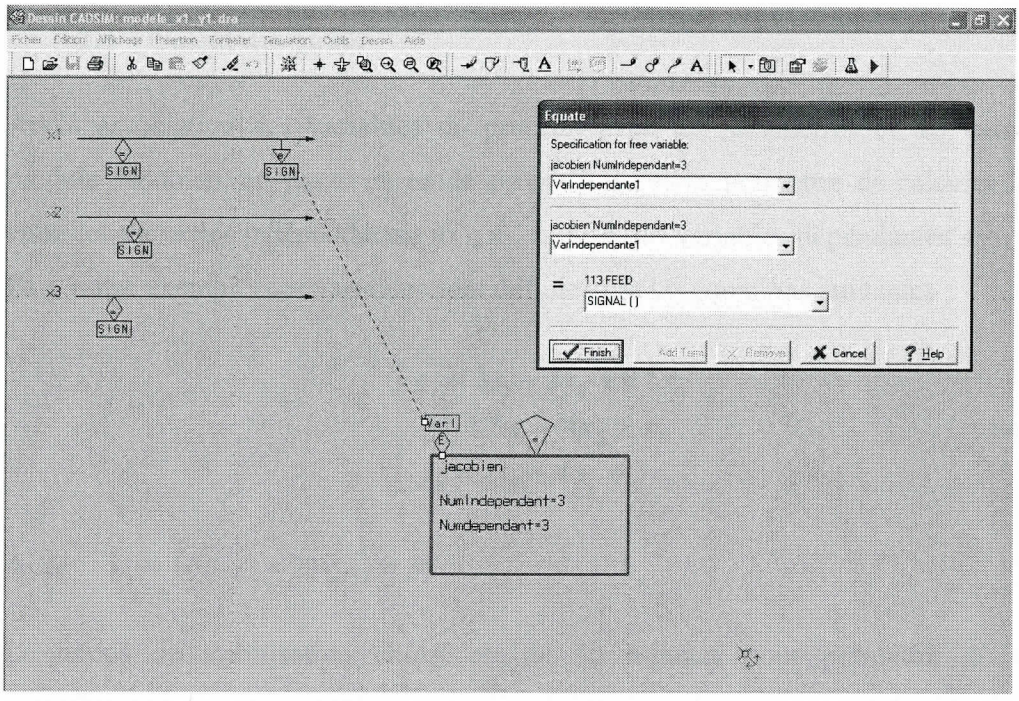


Figure 3.1 Spécification de la variable indépendante 1 au module Jacobien

La même procédure doit être appliquée pour les variables indépendantes 2 et 3 ainsi qu'aux variables dépendantes 1, 2 et 3 restantes.

Le calcul du gain se fait par une perturbation de la variable indépendante avec un facteur de perturbation donné, le module Jacobien calcule la valeur actuelle de la variable indépendante perturbée (CurrentIndependantVariable). Pendant la perturbation du module Jacobien, la conduite (de type signal) «x_1» doit absolument prendre sa valeur de «CurrentIndependantVariable1» du module Jacobien. La conduite «x_1» peut avoir deux valeurs différentes, une à partir de la spécification de base et l'autre venant du module Jacobien lors de la perturbation («Bump» = 1). L'utilisateur doit donc mettre une condition sur la valeur de la variable indépendante «x_1» pour qu'elle soit remplacée par «CurrentIndependantVariable1» lorsque la valeur du paramètre «Bump» est à 1. Sans cette condition, la valeur de la variable «x_1» restera constante même si on procède à une perturbation et donc aucun gain ne sera calculé. La même procédure doit être utilisée pour toutes les variables indépendantes restantes (x_2 et x_3).

3.2.4 Calcul des gains

Après association des variables du procédé avec les paramètres et les variables du module Jacobien, on passe en mode simulation qui va permettre de calculer les gains entre les variables indépendantes «x_1, x_2, x_3 » et les variables dépendantes «y_1, y_2, y_3 ». Pour notre exemple, ces variables sont définies par les équations suivantes :

$$\begin{cases} y_1 = 3x_1 + 2x_2 + 0 * x_3 \\ y_2 = 3x_1 + 4x_2 + x_3 \\ y_3 = 2x_1 + 4x_2 + 3x_3 \end{cases} \qquad \text{Éq. 3.3}$$

Avec : $x_1 = 16$, $x_2 = 23$, $x_3 = 36$

Le temps de stabilisation choisi est de 50 minutes pour permettre de voir les perturbations sur les graphiques qui suivent, malgré qu'en réalité les variables dépendantes «y_i» de cet exemple atteignent leur valeur du régime permanent instantanément (aucune dynamique dans l'équation 3.3).

La figure 3.2 montre la chronologie de la perturbation des variables indépendantes «x_j». Le module Jacobien commence par perturber la valeur de la première variable indépendante x_1 de 10%, pendant le temps de stabilisation de 50 minutes puis calcule les gains dy_1/dx_1, dy_2/dx_1 et dy_3/dx_1. Ensuite il remet la valeur initiale de «x_1» pendant le même temps de stabilisation ce qui permettra aux variables dépendantes de revenir à leurs valeurs initiales. On assume donc ici un procédé monotone où une perturbation inverse à une perturbation initiale pendant le même temps ramène le procédé dans son état initial.

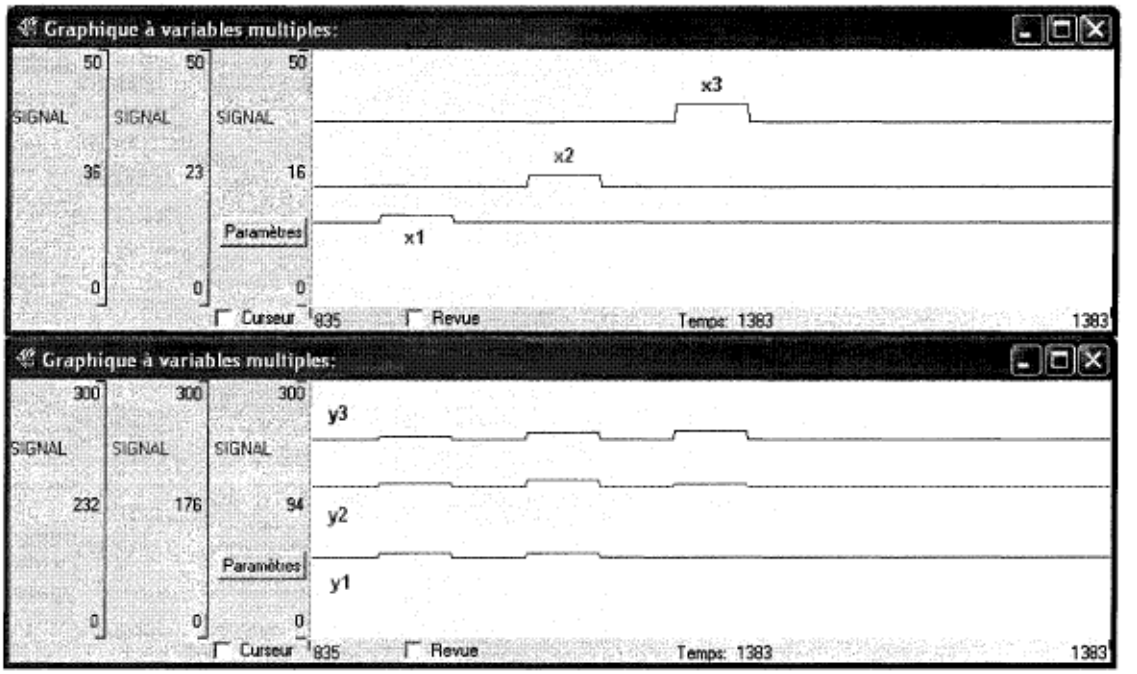


Figure 3.2 **Réponse des variables dépendantes y_i aux perturbations des variables indépendantes x_j après un «Bump» du module Jacobien**

La même logique de perturbation est utilisée pour les variables indépendantes x_2 et x_3, ce qui permet à la fin de calculer les gains entre les variables y_i et x_j (figure 3.3).

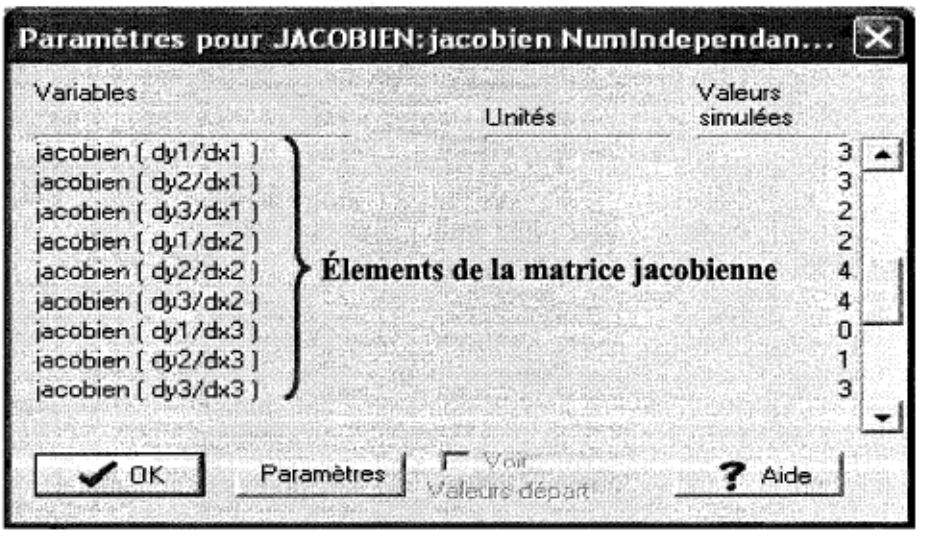


Figure 3.3 **Calcul des gains (dérivées partielles) par le module Jacobien**

Les résultats obtenus au niveau des gains (dérivées partielles dy_i/dx_j) à partir du module Jacobien correspondent bien aux coefficients des équations linéaires qui lient les variables y_i et x_j . Les gains calculés sont des éléments de la matrice jacobienne du système d'équations linéaires présenté par l'équation 3.3.

3.2.5 Conception et programmation du module

L'architecture du logiciel de simulation Cadsim Plus permet aux utilisateurs de créer leur propre module ou unité de procédé avec les fonctionnalités souhaitées. Les propriétés du module sont programmées dans un DLM avec le langage de programmation Borland C++. Des exemples «Templates» sont à la disposition de l'utilisateur pour faciliter la tâche de programmation. Lorsque le DLM est exécuté, le module est automatiquement reconnu par le logiciel, ce qui permet à l'utilisateur d'utiliser toutes les fonctionnalités de Cadsim Plus [26].

L'Annexe 2 montre le listing du programme en Borland C++ utilisé dans le développement du nouveau module Jacobien dans le logiciel Cadsim Plus.

Chapitre 4 - Contrôle d'un concentrateur de liquer noire

4.1 Description de la simulation

La teneur en solides dissous de la liqueur noire en provenance des évaporateurs à multiples effets doit être portée à une valeur comprise entre 65% et 70% dans un concentrateur de liqueur noire pour la combustion dans la chaudière de récupération.

La figure 4.1 présente une simulation d'un concentrateur à circulation forcée de liqueur noire, avec un débit d'alimentation de 58 t/h et 43% de solides dissous qui est préchauffée avec de la vapeur (point 2) dans un échangeur de chaleur (point 1). La liqueur à haute température et pression est admise dans un séparateur de type «flash» (point 3) où la pression est réduite pour évaporer instantanément une certaine quantité d'eau ce qui permet d'augmenter la concentration des solides dissous. La pompe a pour objectif d'augmenter la vitesse de circulation et la pression de la liqueur pour éviter les problèmes d'encrassement et l'ébullition dans l'échangeur de chaleur. Le deuxième échangeur (point 4) permet la condensation de la vapeur à la sortie du séparateur.

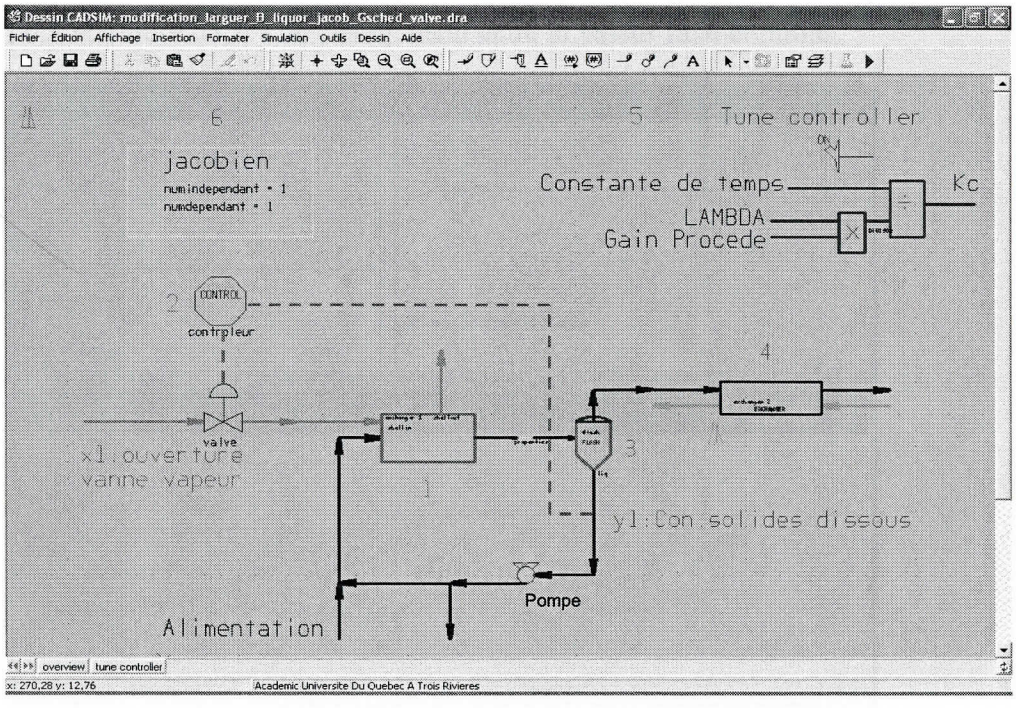


Figure 4.1 Simulation d'un concentrateur de liqueur noire

Un contrôleur de type PI (point 2) est implanté pour un contrôle de la concentration des solides dissous souhaités à la sortie du séparateur, il reçoit son gain Kc à partir d'une logique (point 5) qui utilise la méthode Lambda et le gain de procédé Kp qui provient du module Jacobien (point 6). L'objectif du contrôle est d'atteindre la consigne de solides dissous y_1 en manipulant l'ouverture de la vanne de vapeur x_1.

L'intérêt de cette simulation est de montrer l'effet de la non-linéarité sur la performance du contrôleur et l'utilité du Jacobien pour réajuster le gain du contrôleur.

4.2 Ajustement des paramètres du contrôleur

Pour ajuster les paramètres du contrôleur avec la méthode Lambda, le contrôleur PI est mis en mode manuel autour du point de consigne à 70 % s.d., puis on effectue une perturbation échelon pour caractériser la réponse du procédé avec le module Jacobien.

La figure 4.2 présente la réponse du procédé suite à un changement «instantané» de 5% de la sortie du contrôleur. On constate un comportement de premier ordre, la constante de temps est extraite du graphique de la réponse, elle correspond au temps nécessaire pour atteindre 63,2% du changement entre la valeur final et initial.

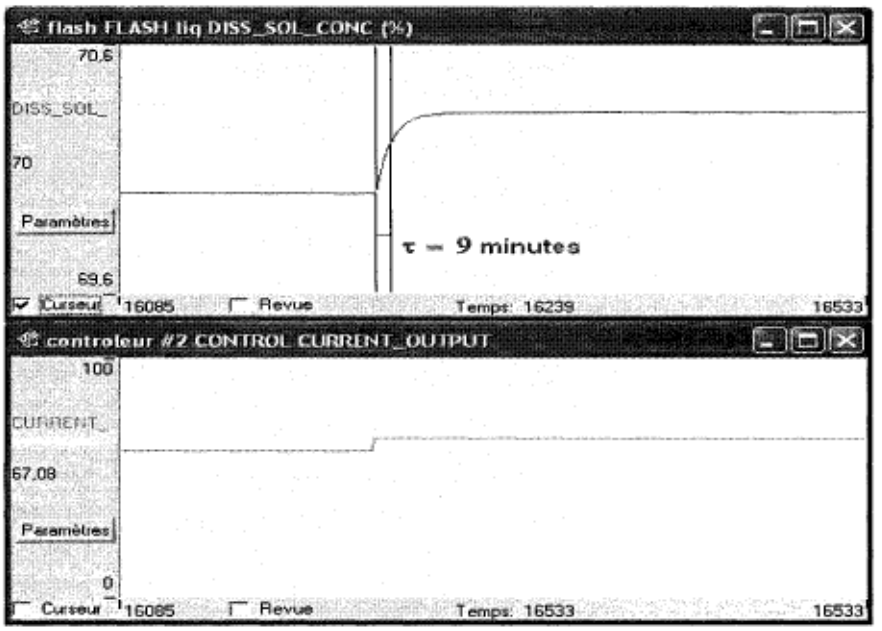


Figure 4.2 Réponse du procédé de premier ordre.

Le gain du procédé K_p est automatiquement calculé par le module Jacobien lorsque l'utilisateur demande une perturbation comme le montre la figure 4.3 :

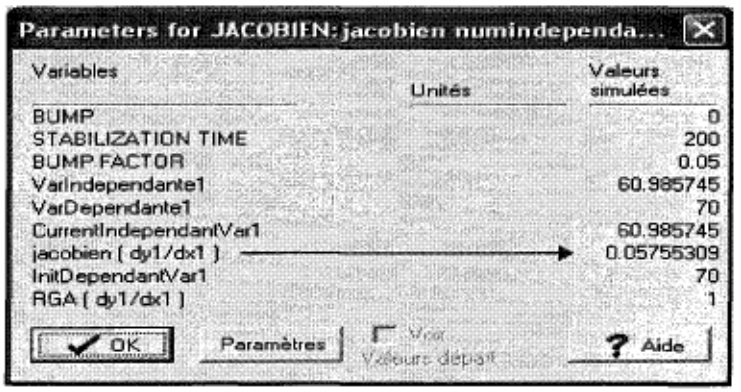


Figure 4.3 Valeur calculée du module Jacobien

$$K_p = 0.0576 \frac{\%\ solides\ dissous}{\%\ ouverture\ de\ vanne} \qquad \text{Éq. 4.1}$$

Sachant que la dynamique du procédé suit un comportement de premier ordre, on peut alors calculer les paramètres du contrôleur (Annexe 1).

Le gain proportionnel K_c est calculé par l'équation suivante :

$$K_c = \frac{\tau}{K_p \lambda} \qquad \text{Éq. 4.2}$$

Il faut choisir le paramètre Lambda de façon à avoir une réponse rapide sans «dépassement» de la variable contrôlée y_1 suite à un changement de point de consigne. La valeur de Lambda choisit après plusieurs essai est de 10 minutes.

$$K_c = \frac{9}{0{,}0576 * 10} = 15{,}63$$

La constante de temps intégrale τ_I est équivalente à la constante de temps du procédé :

$$\tau_I = \tau = 9 \text{ minutes}$$

La figure 4.4 montre la réponse du contrôleur après un changement de point de consigne de 70% à 68 % de solides dissous. Le contrôle se fait sans oscillation ni dépassement de la variable contrôlée. On peut dire que le contrôleur est bien ajusté dans cette plage d'opération.

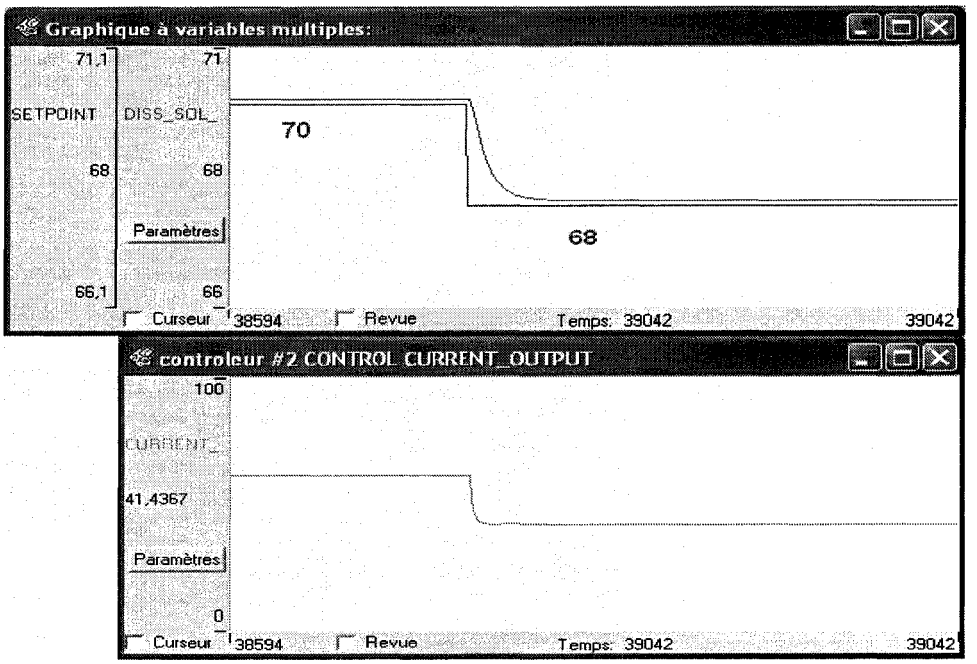


Figure 4.4 Réponse du contrôleur après un changement du point de consigne de 70% à 68 % de solides dissous.

4.2.1 Effet de la non-linéarité sur la performance du contrôleur

Lorsque qu'on demande un changement de point de consigne à 68% puis à 66 % des solides dissous, la réponse du contrôleur est satisfaisante. Par contre, lorsqu'on passe à 64% de point de consigne, on constate un «dépassement» de la sortie du contrôleur, cette situation s'empire à 62% des solides dissous avec une oscillation de la variable contrôlée. À 60% la situation est totalement hors contrôle et le contrôleur est instable ce qui provoque une oscillation permanente de la variable contrôlée «y_1» (voir figure 4.5).

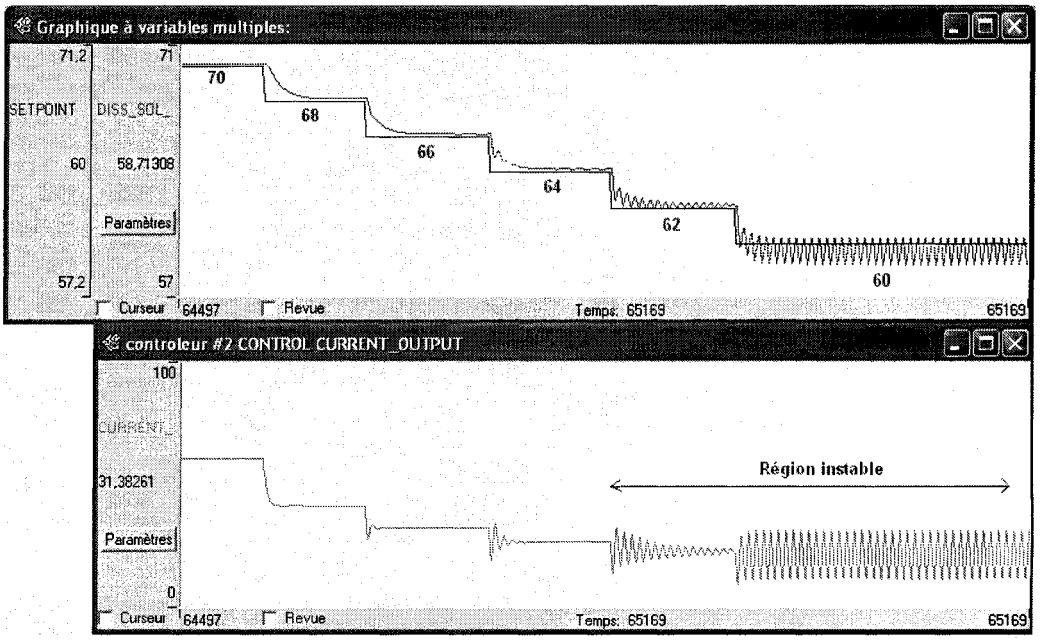


Figure 4.5 Effet du changement du point de consigne sur la performance du contrôleur.

Cette perte de performance du contrôleur peut être expliquée par la non-linéarité du procédé. Pour remédier à cette situation, il faudrait investiguer les changements au niveau des gains lors des changement des conditions d'opération (tableau 4.1).

4.2.2 Changements de gain sur la plage d'opération

Le tableau 4.1 montre très bien un grand changement au niveau du gain qui sont de 3 à 12 fois plus important que le gain calculé à 70 % de concentration des solides dissous. Le gain du contrôleur Kc calculé à 70 % est trop élevé pour l'utiliser dans la plage d'opération de 64 % à 60 %.

Tableau 4.1 Les changements au niveau des gains selon le point d'opération.

Point de consigne	70 %	68 %	66 %	64 %	62 %	60 %
Gain Kp (dy1/dx1)	0,0575	0,1549	0,2724	0,4	0,5357	0,6683
Kp / Kp(à 70%)	1,0	2,7	4,7	7	9,3	11,6

Pour expliquer les résultats du tableau 4.1, la figure 4.6 présente la variation du débit en fonction de l'ouverture de la vanne. On voit clairement que le gain de la valve (pente de la courbe) dans la région d'opération de 64% à 60% est très élevé par rapport à la région 70 % à 68 %. Ceci est dû aux caractéristiques de la valve utilisée dans la simulation qui est de type «quick opening» où la majeure variation du débit (80%) s'effectue dans une plage restreinte d'ouverture de la valve (premier 20% d'ouverture). Ce type de valve n'est pas recommandable pour contrôler le débit, mais le but de la simulation est de montrer l'impact négatif de la non-linéarité sur la performance du contrôleur, c'est pourquoi ce type de valve est utilisé ici.

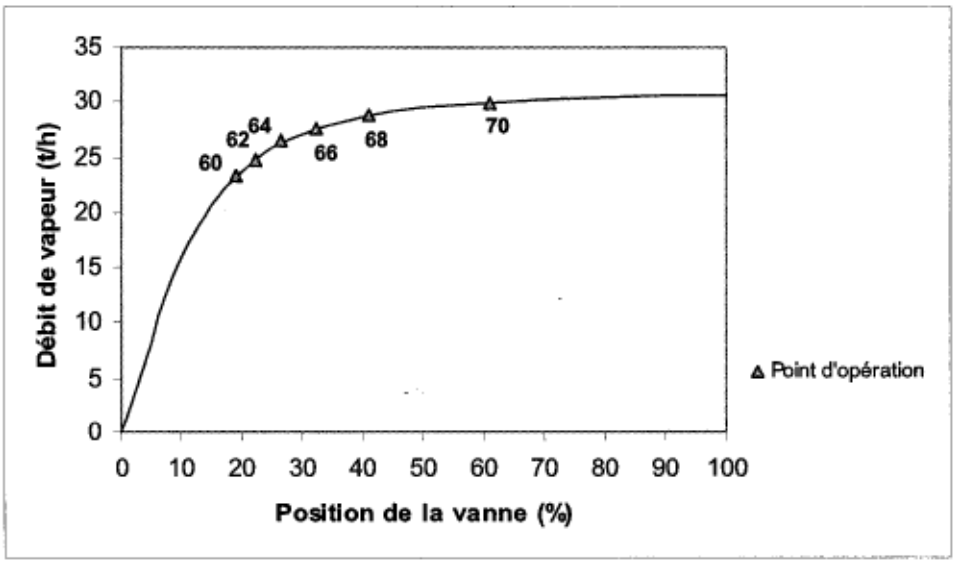


Figure 4.6 Évolution du gain de la valve sur la plage d'opération.

4.3 Utilité du module Jacobien

Pour un ajustement optimal du gain du contrôleur Kc, une logique est introduite dans la simulation qui permet à l'utilisateur d'ajuster automatiquement le gain à partir du module Jacobien en demandant un test d'échelon (figure 4.2). La constante de temps du procédé et le paramètre Lambda sont introduits manuellement et restent inchangés. Le gain du procédé prend sa valeur du module Jacobien. Le gain calculé du contrôleur se fait avec la méthode Lambda.

La figure 4.7 montre comment est ajusté le gain du contrôleur au point d'opération de 60% de solides dissous lorsque la réponse se met à osciller. La première étape est de mettre le contrôleur en mode manuel et ensuite demander une perturbation de type échelon au module Jacobien, ce qui permettra de calculer le gain optimal du procédé à 60 % des solides dissous. Le gain du contrôleur est modifié automatiquement et il vaut 1,71 et donc neuf fois inférieur à la valeur initiale de 15,63.

Maintenant que le contrôleur est ajusté, il est mis en automatique pour pouvoir effectuer les changements de point de consigne autour du point d'opération à 60% et à 62 % où l'on constate une réponse sans oscillation de la variable contrôlée. L'utilisateur peut suivre les mêmes étapes pour ajuster le contrôleur dans une plage d'opération spécifique.

A noter que l'on doit ajuster le contrôleur à chaque nouveau point d'opération puisque le module garde en mémoire la dernière valeur du gain. Si on revient à 70 % de solides dissous comme point de consigne, l'utilisateur devrait demander au module Jacobien de refaire le test échelon.

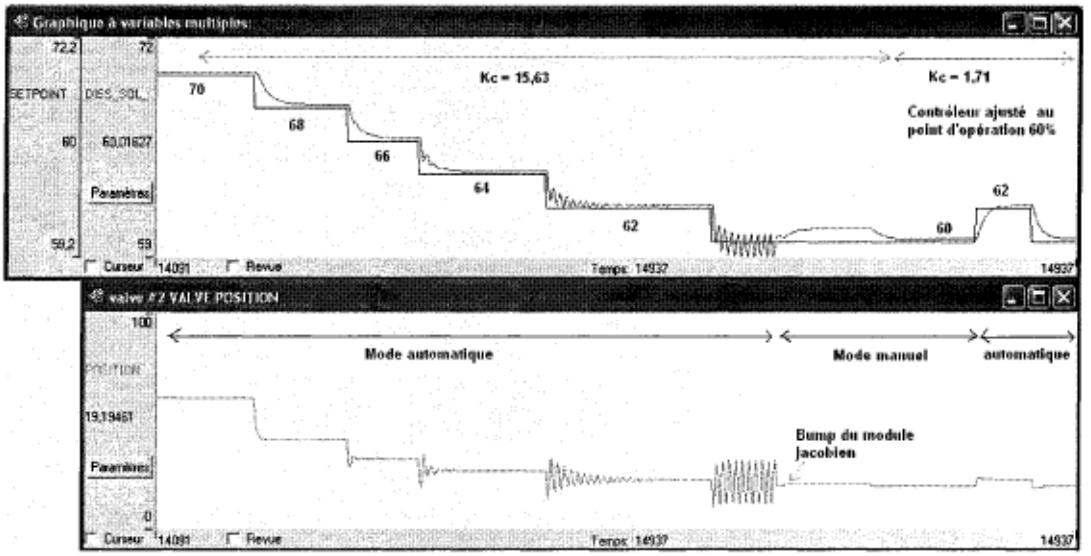


Figure 4.7 Ajustement du gain du contrôleur à partir du module Jacobien.

4.4 Conclusions

Les résultats de la simulation de la liqueur noire montrent que le module Jacobien est capable de réajuster le gain du contrôleur lorsque l'utilisateur en fait la demande. L'effet de la non-linéarité causé par la valve augmente l'instabilité du contrôleur sur la plage d'opération de 70% à 60% de la concentration des solides dissous. Le module Jacobien a permis une stabilisation du contrôle des solides dissous avec l'introduction du gain de contrôleur optimal au point d'opération désiré.

Chapitre 5 - Contrôle du lavage de la pâte chimique

Dans la librairie des simulations du logiciel Cadsim Plus on trouve des exemples utilisés dans le domaine des pâtes et papiers. Le répertoire «Samples» contient en effet beaucoup d'exemples comme une simulation de lavage de pâte d'une usine kraft [27] avec l'optimisateur de l'eau de lavage.

La simulation de l'optimisateur de lavage de pâte est très intéressante, elle permet de trouver le facteur de dilution optimal pour un débit donné en bois mou «Softwood». Ce facteur de dilution optimal correspond au coût minimum d'opération en tenant compte des coûts liés aux pertes des produits chimiques de cuisson, utilisation excessive des produits lors du blanchiment ainsi qu'à l'évaporation [28].

Cette simulation est utilisée ici pour faire un contrôle de lavage efficace et optimal en utilisant le module Jacobien développé.

5.1 Description de la simulation

Le lavage de la pâte cuite en provenance du lessiveur est une opération très importante qui sert à séparer les fibres des composants organiques (lignine) et les produits chimiques inorganiques utilisés lors de la cuisson.

La figure 5.1 représente un système de lavage et classage de la pâte brune dans une usine kraft moderne qui inclus une étape de délignification à l'oxygène.

À la sortie du lessiveur (point 1) en continu, la pâte passe par deux stades de lavage par diffusion (point 2) se caractérisant par un contact entre la pâte et la liqueur de lavage pour faciliter la diffusion des matières dissoutes. La pâte lavée à la sortie des diffuseurs est admise dans un réservoir de stockage à haute densité (point 3) qui alimente les trieurs de nœuds (point 4) et les classeurs (point 5) pour séparer les copeaux non cuits et les bûchettes de la pâte.

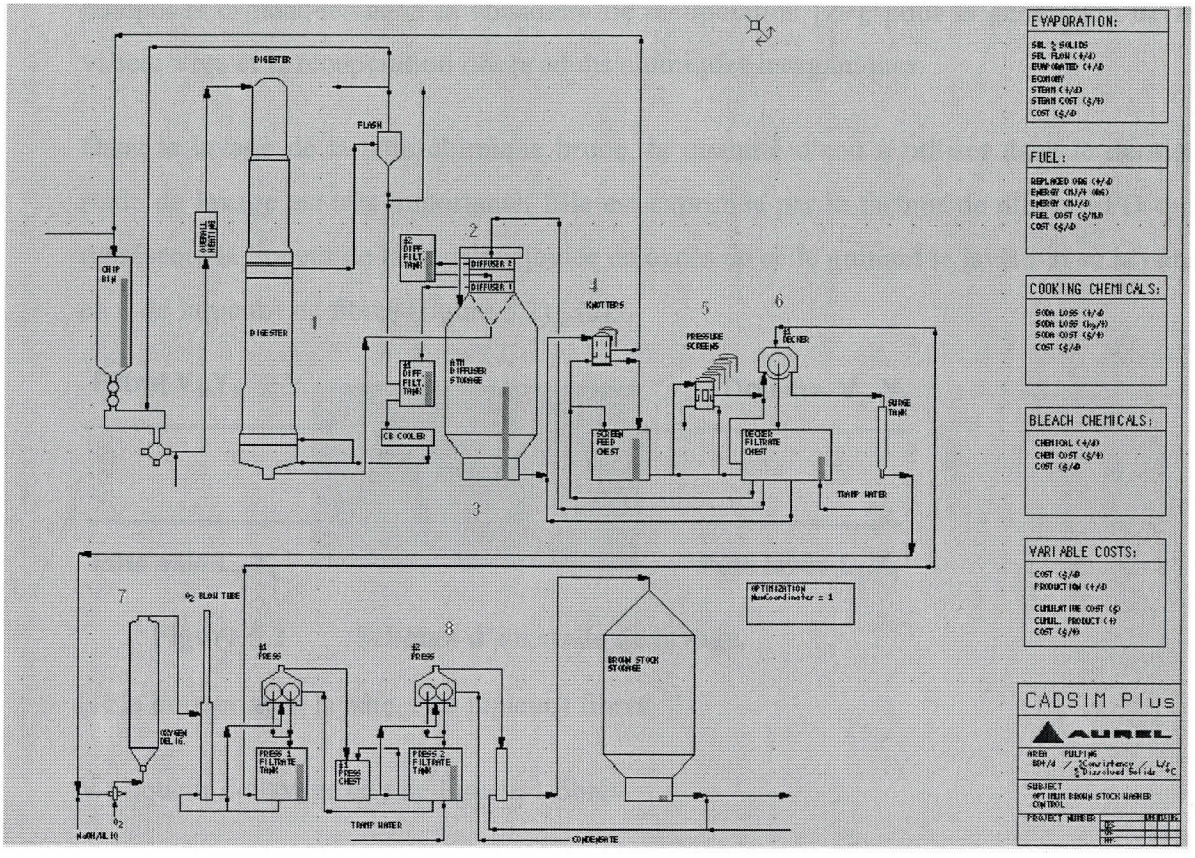


Figure 5.1 Simulation du lavage de pâte Kraft

La pâte acceptée des classeurs est lavée et épaissiE à une consistance de 14% dans une pile laveuse (point 6) avec application de vide puis admise dans un réacteur de délignification à l'oxygène (point 7) dans le but d'enlever le maximum de lignine encore présente (2% du bois mou en base sèche est solubilisé à cette étape) pour réduire l'utilisation des produits chimiques lors du blanchiment en plus de réduire la pollution qui en résulte.

Finalement la pâte est lavée en deux stades à contre-courant où l'eau fraîche est introduite dans le laveur à presse n°2 (point 8) alors que le filtrat sert d'eau de lavage au stade précédent [28].

La liqueur noire est envoyée aux évaporateurs et ensuite au concentrateur qui augmentent sa concentration à un niveau élevée (60 à 80 %) en vue de la combustion des

composés organiques dans la chaudière de récupération [29], pour la génération de la vapeur vive et la récupération des produits chimiques inorganiques.

Dans le lavage de la pâte chimique brune, la quantité d'eau à utiliser dans le dernier stade de lavage est très importante. Elle est exprimée par le facteur de dilution FD qui représente la liqueur de la douche ajoutée en excès de celle quittant le laveur avec la pâte en t de liqueur/t de fibres (figure 5.2) [30].

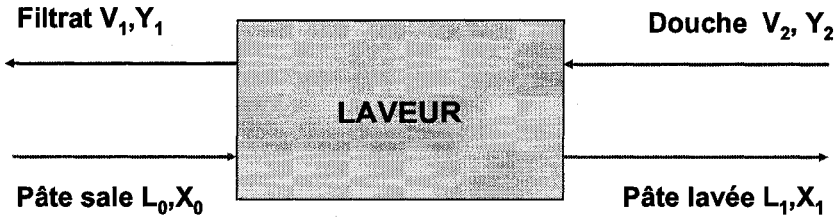


Figure 5.2 Schéma d'un stade de lavage.

L : la liqueur avec la pâte, t de liqueur/t fibres

V : liqueur de lavage en t de liqueur/t fibres

X, Y : concentration des solides dissous en kg/t

$$FD = V_2 - L_1 \qquad \text{Éq. 5.1}$$

Le rapport de déplacement RD permet de comparer la réduction réelle de la teneur en matières solides à la réduction maximale possible :

$$RD = \frac{X_0 - X_1}{X_0 - Y_2} \qquad \text{Éq. 5.2}$$

5.2 Composantes du coût d'opération de l'usine

Les objectifs du lavage de pâte brune sont :

- Élimination de la liqueur résiduelle qui pourrait contaminer la pâte dans les étapes de traitement subséquentes [29].

- ❖ Récupération des produits chimiques de cuisson avec un minimum de dilution
- ❖ Réduction de la consommation des produits chimiques lors du blanchiment [31].
- ❖ Minimisation des effluents liquides et réduction de la pollution de l'eau et de l'air [30].

Ces différents objectifs paraissent un peu contradictoires du fait qu'un bon lavage requiert l'utilisation de grosses quantités d'eau et que de l'autre côté il faut minimiser la dilution pour ne pas surcharger la capacité des évaporateurs [30]. Néanmoins il existe un facteur de dilution optimal correspondant à la quantité d'eau à utiliser pour minimiser les coûts d'opération. Ce facteur de dilution varie d'une usine à l'autre et il est lié aux coûts des produits chimiques de cuisson, de blanchiment et d'évaporation.

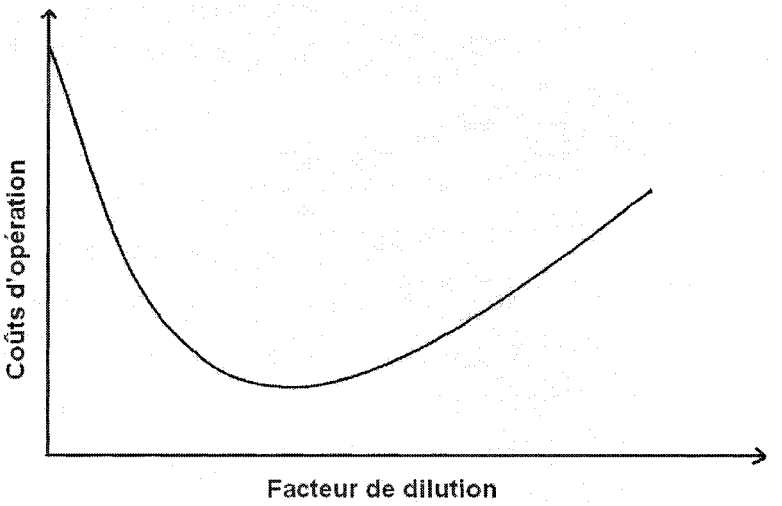


Figure 5.3 Facteur de dilution optimal.

La figure 5.3 indique qu'il existe un point optimal où les coûts d'opérations sont minimisés [28]. La simulation permet de trouver ce point à l'aide de l'optimisateur qui essaie différents débits d'eau de lavage au dernier stade pour minimiser le coût total d'opération.

La simulation calcule les coûts liés à l'évaporation, aux produits chimiques de la cuisson, à la consommation des produits de blanchiment et aux pertes de matières organiques (lignine, résines) qui auraient pu servir comme combustible. Les modules ou

boîtes empilées à la droite dans la figure 5.1 calculent ces coûts durant la simulation du procédé. Ces coûts sont détaillés dans les sections qui suivent.

5.2.1 Coûts d'évaporation

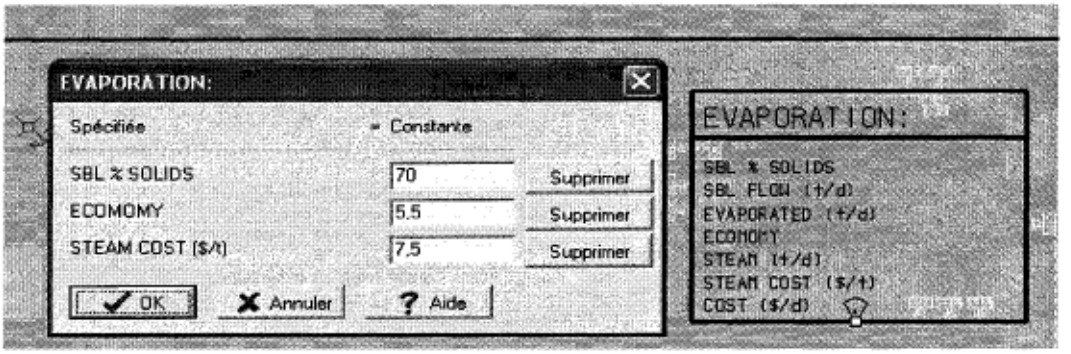


Figure 5.4 Paramètre pour calcul des coûts d'évaporation

La figure 5.4 montre les variables spécifiées qui permettent le calcul des coûts de l'évaporation :

SBL % solids : la concentration finale des solides dissous dans la liqueur concentrée après évaporation.

L'économie : C'est une mesure de l'efficacité des évaporateurs, elle est exprimée en kg d'eau évaporée par kg de vapeur utilisée

Steam cost : le coût en dollars par tonne de vapeur.

On veut savoir la quantité d'eau à évaporer pour que la teneur des solides dissous soit égale à 70% dans la liqueur. Pour cela il faut calculer le débit de liqueur concentrée «SBL flow» qui correspond à la quantité total des solides dissous :

$$\text{SBL Flow (t/d)} = \text{Solides dissous totaux} / \text{SBL \% solids} \qquad \text{Éq. 5.3}$$

La quantité d'eau à évaporer correspond à la différence entre le débit de la liqueur faible (avant l'évaporation) et le débit de la liqueur concentrée :

$$\text{Evaporated (t/d)} = \text{Flow} - \text{SBL Flow} \qquad \text{Éq. 5.4}$$

Connaissant la quantité d'eau à évaporer on peut calculer la quantité de vapeur requise par les évaporateurs :

$$Steam\,(t/d) = \frac{Evaporated\,(t/d)}{Economy} \qquad \text{Éq. 5.5}$$

Finalement le coût de l'évaporation journalier est :

$$Cost_{evap}\ (\$/d) = Steam\ (t/d) * Steam\ Cost\ (\$/t) \qquad \text{Éq. 5.6}$$

5.2.2 Coûts des produits chimiques de cuisson

Ces coûts sont évalués à partir de la quantité irrécupérable de sulfate de sodium présente dans la pâte lavée qui est envoyée à la section de blanchiment. La quantité perdue (Soda Loss) est exprimée par l'équation suivante :

$$Soda\ Loss\ (t/d) = 0.95 * Inorganic\ (t/d) \qquad \text{Éq. 5.7}$$

Le coût journalier des produits chimiques de cuisson perdue :

$$Cost_{cooking}\ (\$/d) = Soda\ Loss\ (t/d) * Soda\ Cost\ (\$/t) \qquad \text{Éq. 5.8}$$

La qualité de lavage est exprimée par la quantité de sulfate de sodium encore présente par rapport à la quantité de pâte sèche :

$$Soda\ Loss\,(kg/t) = \frac{Soda\ Loss\,(t/d)}{Softwood\,(t/d)} * 1000 \qquad \text{Éq. 5.9}$$

5.2.3 Coût des produits chimiques de blanchiment

Lorsque la pâte n'est pas suffisamment lavée, l'étape de blanchiment requiert un apport supplémentaire de produits chimiques lié à la présence des matières organiques tel que la lignine.

La quantité de produits chimiques de blanchiment utilisée est calculée à partir de la quantité total de solides dissous présente dans la pâte quittant le dernier stade de lavage :

$$\text{Chemicals (t/d)} = 0.4 * \text{quantité totale de solides dissous (t/d)} \qquad \text{Éq. 5.10}$$

Le coût journalier des produits de blanchiment requis :

$$\text{Cost}_{\text{chemicals}} (\$/d) = \text{Chemicals (t/d)} * \text{Chemicals Cost (\$/t)} \qquad \text{Éq. 5.11}$$

5.2.4 Coûts de combustible

Ces coûts sont liés aux pertes des produits organiques qui ne sont plus récupérables et qui ont un potentiel énergétique. L'équation utilisée dans la simulation est spécifique pour une consistance de copeaux de 50%, elle ne permet pas d'évaluer les coûts liés au combustible pour une plage d'opération de 45% à 55 %. Étant donné que le coût lié au combustible est plus faible par rapport aux autres coûts d'opération, et presque constant avec la variation du facteur de dilution (voir figure 5.7), il n'est pas pris en compte dans cette simulation de contrôle du lavage.

5.3 Facteur de dilution optimal

La simulation de la figure 5.1 présente le lavage d'une pâte chimique à 50% de consistance avec un débit sec de bois mou équivalent à 2000 t/j. Après le stade de cuisson et de délignification à l'oxygène, le débit sec de la pâte atteint 892 t/j, soit une rendement global de 44,6 %.

La quantité d'eau de lavage à contre courant qui est directement liée au facteur de dilution, a un impact majeur sur les coûts d'évaporation et des produits chimiques comme le montre les figures 5.5 et 5.6.

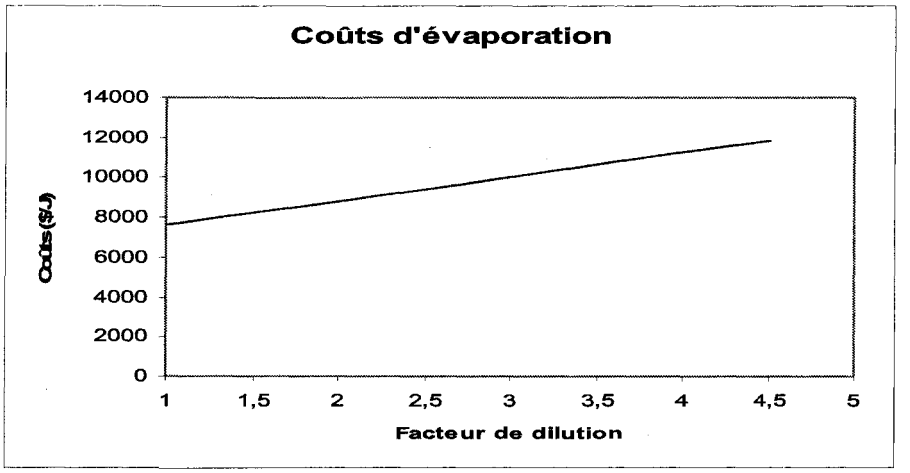


Figure 5.5 Coûts d'évaporation en fonction du facteur de dilution.

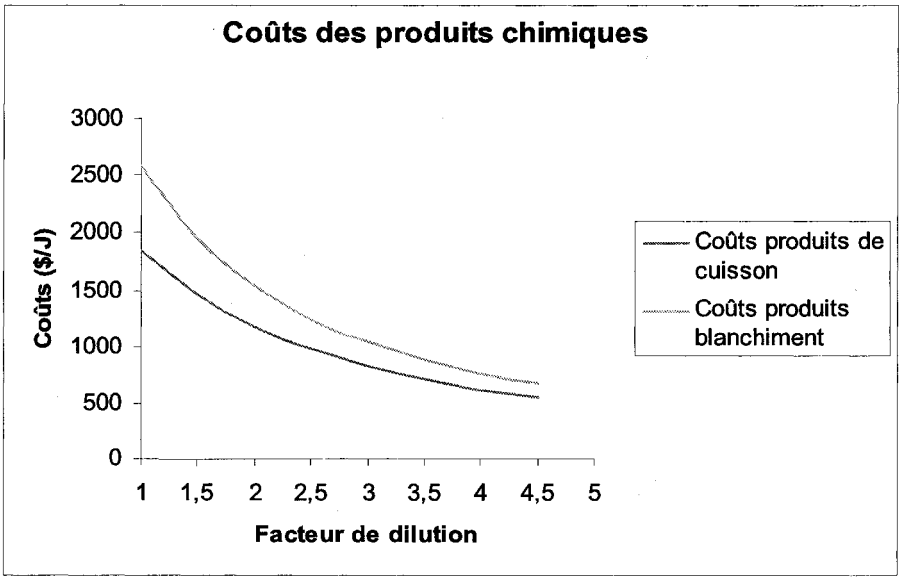


Figure 5.6 Coûts des produits chimiques.

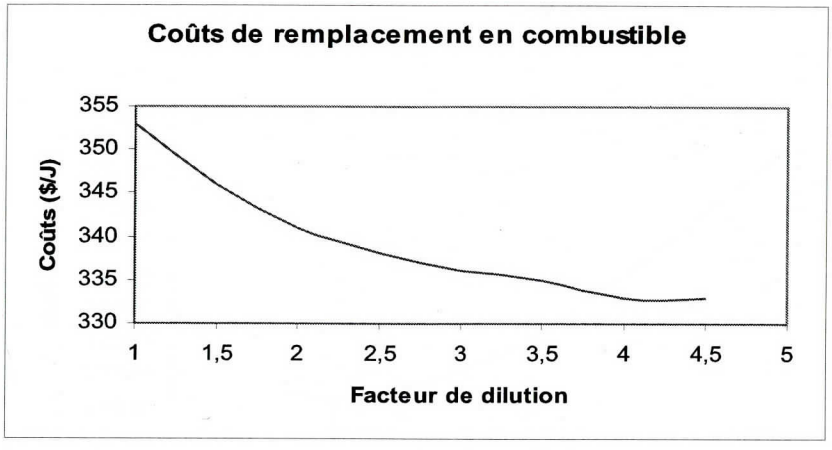


Figure 5.7 Coûts en remplacement en combustible.

Comme on peut le constater, l'évaporation présente un pourcentage important (74%) du coût total de lavage, car elle est directement liée à la quantité d'eau introduite dans le système (figure 5.8).

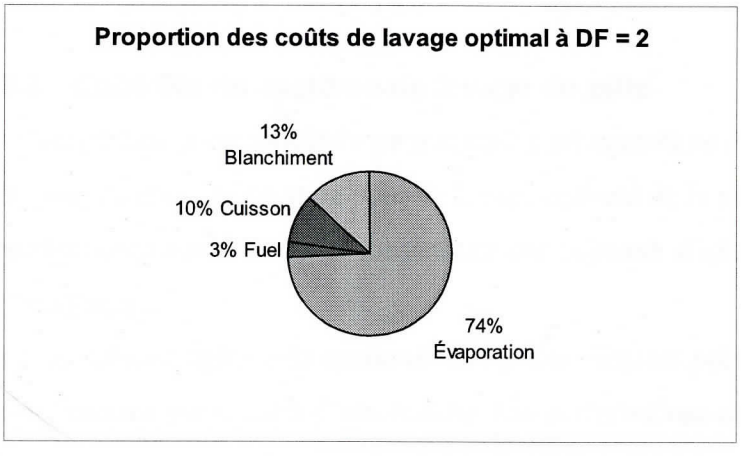


Figure 5.8 Répartition des coûts d'opération.

Le facteur de dilution optimal correspond à la quantité d'eau de lavage qui minimisera les coûts totaux d'opération comme le montre la figure 5.9. La simulation de lavage à 2000 t/j de bois mou calcule un facteur de dilution optimal au dernier stade de lavage à 1,95 soit un débit d'eau fraîche de 4429 t/j.

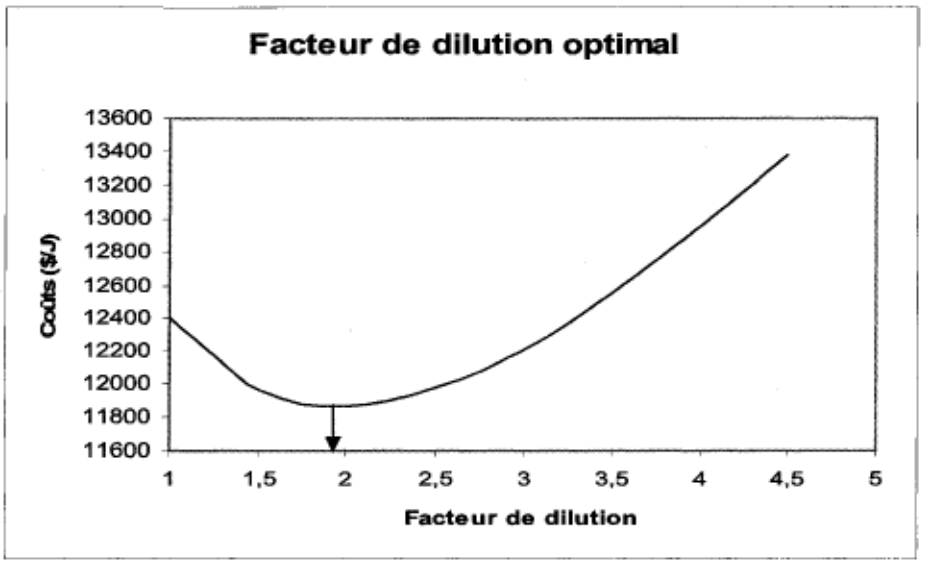


Figure 5.9 **Facteur de dilution optimal à 2000t/j et une consistance de 50%.**

5.4 Contrôle du système de lavage de pâte

La simulation a été modifiée pour installer un contrôleur PI qui permettra de manipuler la quantité d'eau nécessaire pour un lavage optimal de la pâte dans une plage variable de production au niveau de la consistance des copeaux d'alimentation (de 45% à 55% de consistance).

Le contrôleur mesure la quantité de solides dissous présente dans la pâte et agit en conséquence sur le débit d'eau fraîche. Les perturbations au niveau de la consistance des copeaux ont une influence sur la performance du contrôleur. Pour cette raison, les paramètres du contrôleur doivent être ajusté en utilisant une stratégie de programmation des gains.

5.4.1 Système de contrôle

La figure 5.10 montre la disposition du contrôleur PI qui contrôle la concentration de solides dissous (kg sd/t pâte) à la sortie du dernier laveur «y_1» avec la variable manipulée «x_1» qui est la quantité d'eau de lavage (l/s) introduite au dernier laveur (Press #2).

La quantité de solides dissous total par tonne de pâte sèche est calculée par l'addition du «soda loss» et la quantité des organiques en kg/t.

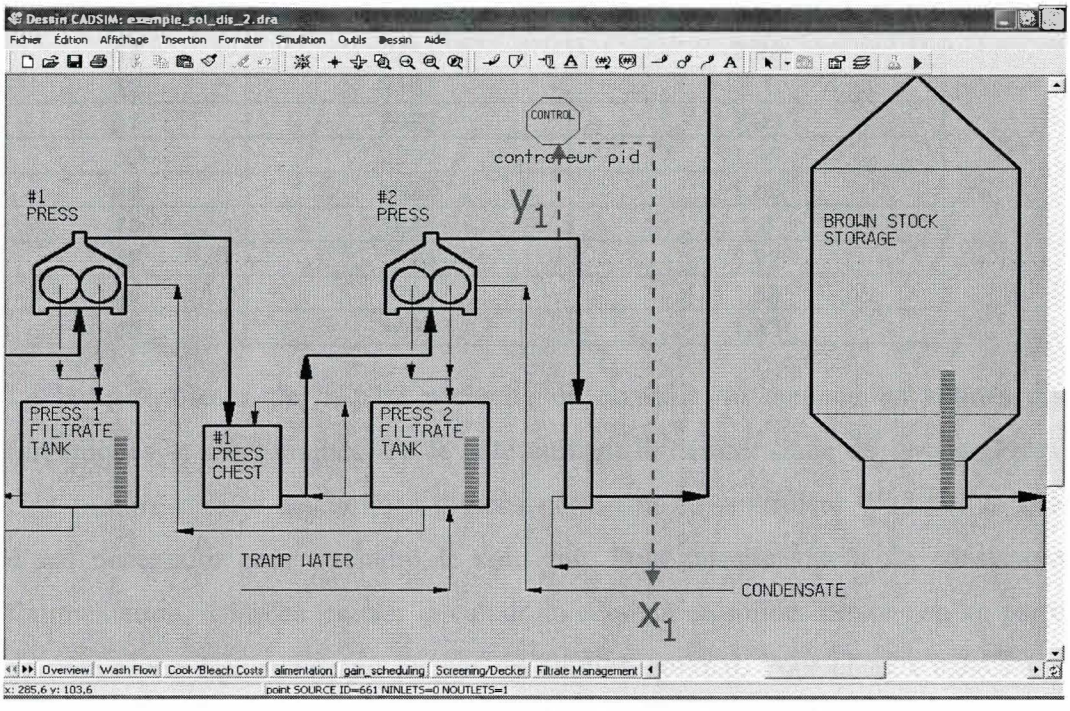


Figure 5.10 Contrôleur des solides dissous.

5.4.2 Point de consigne optimal de solides dissous

Les changements au niveau de la consistance affectent les coûts d'opération et donc le facteur de dilution optimal. À partir de la simulation de l'optimisateur de lavage, le tableau 5.1 montre le facteur de dilution optimal équivalent à chaque consistance. On rappelle que la quantité d'eau présente dans les copeaux est constante à 2000t/j et que les coûts en combustible ne sont pas pris en compte pour les raisons expliquées auparavant.

Tableau 5.1 Facteur de dilution optimal en fonction de la consistance

Consistance (%)	débit sec bois mou (t/j)	Facteur de dilution optimal
55	2444	2,3
54	2348	2,24
53	2255	2,22
52	2167	2,1
51	2082	2,03
50	2000	1,95
49	1922	1,88
48	1846	1,8
47	1774	1,72
46	1704	1,63
45	1636	1,53

Le lavage efficace de la pâte nécessite un contrôle du contenu en solides dissous (organiques et inorganiques) de la pâte quittant le dernier stade de lavage (Press #2). Une mesure directe des solides dissous permettra au contrôleur d'ajuster la quantité d'eau nécessaire pour atteindre la consigne. Pour chaque valeur de consistance de l'alimentation, il faudra trouver la valeur du contenu en solides dissous en kg par tonne de pâte sèche équivalente au facteur de dilution optimal. Pour chaque consistance, le contrôleur est mis en mode manuel et on change la valeur du débit d'eau fraîche jusqu'à ce que le facteur de dilution optimal soit atteint. On note la valeur des solides dissous en kg/t de pâte équivalente à la somme des contenus organiques et le «soda loss» (tableau 5.2).

Tableau 5.2 Contenu en solides dissous vs facteur de dilution optimal

Consistance (%)	Facteur de dilution optimal	Solides dissous kg/t
55	2,30	17,08
54	2,24	16,68
53	2,17	16,60
52	2,10	16,52
51	2,03	16,43
50	1,95	16,42
49	1,88	16,28
48	1,80	16,21
47	1,72	16,18
46	1,63	16,16
45	1,53	16,15

Pour la plage d'opération de la consistance, la moyenne des contenus en solides dissous est équivalente à 16,4 kg/t avec un «soda loss» de 10,3 kg/t. Un lavage efficace de la pâte équivaut à un contenu de «soda loss» autour de 10 kg /t [32]. Le point de consigne choisit obéit au critère économique du coût minimal et au critère de qualité de la pâte lavée.

Avec cette valeur du point de consigne, le contrôleur peut varier son débit d'eau fraîche pour un lavage optimal de la pâte. Pour confirmer le choix de la valeur du point de consigne des solides dissous à 16,4 kg /t le tableau 5.3 mesure l'écart entre le facteur de dilution effectif et le facteur de dilution optimal.

Tableau 5.3 Écart entre les facteurs de dilution effectif et optimal

Consistance (%)	Facteur de dilution optimal	Facteur de dilution effectif	erreur relative
55	2,30	2,40	-4,35%
54	2,24	2,28	-1,79%
53	2,17	2,20	-1,38%
52	2,10	2,12	-0,95%
51	2,03	2,04	-0,49%
50	1,95	1,95	0,00%
49	1,88	1,87	0,53%
48	1,80	1,78	1,11%
47	1,72	1,69	1,74%
46	1,63	1,60	1,84%
45	1,53	1,50	1,96%

5.4.3 Ajustement des paramètres du contrôleur

Comme mentionné auparavant, dans l'industrie papetière, la méthode Lambda est très utilisée car elle offre un contrôle avec peu d'oscillation et de dépassement ce qui a pour effet une réduction de la variabilité [8].

5.4.3.1 Paramètres du module Jacobien

Pour ajuster les paramètres du contrôleur avec la méthode Lambda, il faut calculer le gain du procédé et la constante de temps suite à une excitation de la variable manipulée qui correspond à la quantité d'eau fraîche. L'utilisateur peut choisir soit d'augmenter ou

de diminuer la valeur de la variable manipulée avec un certain pourcentage en utilisant un «Perturbation factor» positif ou négatif. Sachant qu'une augmentation de la quantité d'eau diminue la quantité des solides dissous, un facteur de perturbation négatif du module Jacobien à -0,1 (-10%) a été utilisé pour avoir une réponse standard.

Le temps de stabilisation est très important puisqu'il influence directement la valeur du gain du procédé. La variable mesurée (dépendante) doit atteindre le régime permanent avant d'effectuer le calcul du gain. En effectuant une perturbation de la variable manipulée (figure 4.20), il faut à peu près de 4000 minutes (≈ 3 jours) pour atteindre le régime permanent au niveau des solides dissous mesurés. Le choix d'une valeur de 5000 itérations (1 itération /minute) pour le temps de stabilisation est donc largement suffisant pour effectuer le calcul du gain.

5.4.3.2 Paramètres de la réponse dynamique du procédé

Maintenant que les paramètres du module Jacobien sont soigneusement choisis, on peut alors effectuer le test (figure 5.11) pour un calcul automatique du gain du procédé et déterminer la constante de temps et la nature de la réponse à partir du graphique de la figure 4.20.

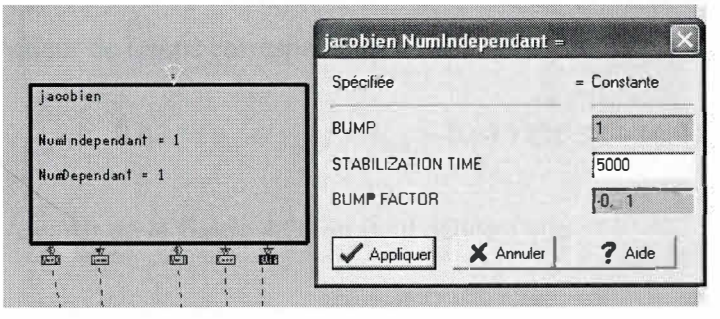


Figure 5.11 Paramètres du module Jacobien

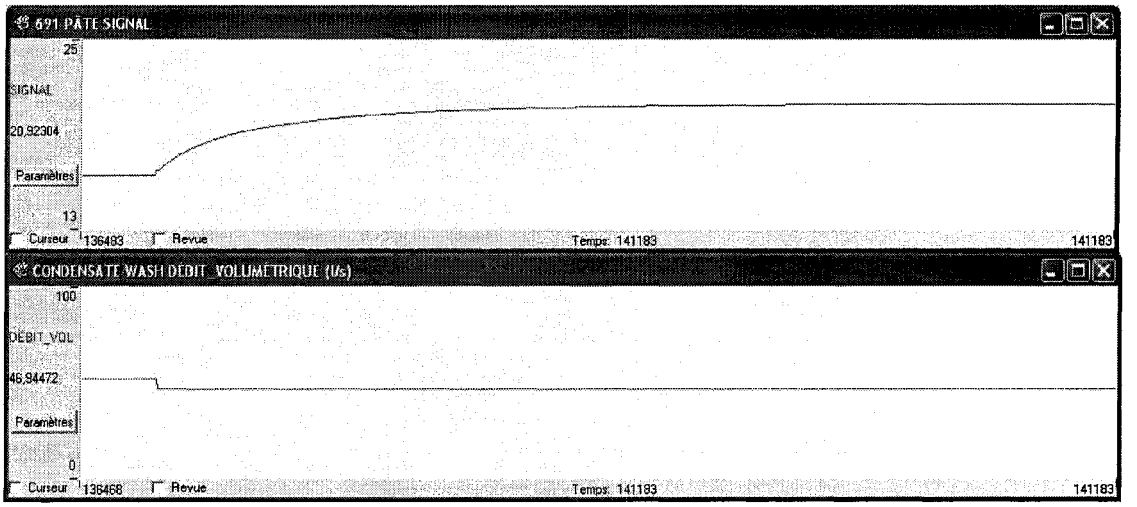


Figure 5.12 Réponse dynamique à la perturbation du module Jacobien

La figure 5.12 montre le résultat de la perturbation du module Jacobien où on a diminué le débit volumétrique de 10% (de 52,16 à 46,94 L/s), il en résulte une augmentation du contenu en solides dissous de 28% (de 16,4 à 20,92 kg/t pâte sèche). Le gain du procédé calculé par le module Jacobien est de -0.867 (kg/t / L/s) (figure 5.13).

La réponse dynamique est du premier ordre sans délai, la constante de temps est déduite à partir du temps nécessaire pour atteindre 63,2 % de la réponse. Il faut alors chercher la valeur de temps correspondant à :

$$Y_{63\%} = \quad 0.63 * (y_{\infty} - y_{init}) + y_{init} = 0.63 * (20.92 - 16{,}4) + 16{,}4 = 19{,}24\,\text{kg/t}\,.$$

À partir de la figure 4.20 on peut déduire que :

$\tau = 300$ min = 5h

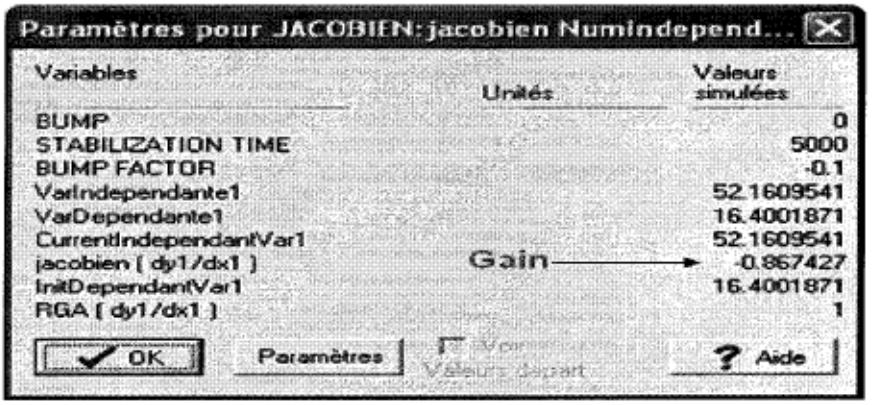


Figure 5.13 Gain calculé par le module Jacobien

5.4.3.3 Paramètres liés à l'ajustement par la méthode Lambda

Les paramètres du contrôleur (gain et temps intégral) sont calculés avec la méthode Lambda (voir Annexe1). Le facteur Lambda est choisit par essai et erreur jusqu'à l'obtention d'une réponse stable et rapide. Une valeur de Lambda de 60 min est admise, alors :

Le gain du contrôleur calculé à partir de l'équation 4.2 est :

$$K_c = \frac{\tau}{K_p \lambda} = \frac{300}{-0{,}867 * 60} = -5.76$$

Puisque la constante de temps et trop longue, la méthode de Skogestad limite la valeur de la constante intégrale τ_I [6]:

$$\tau_I = \min\{ \tau , 4 (\lambda + \theta) \} = 240 \text{ min} \qquad \text{Éq. 5.12}$$

Pour une réponse plus rapide, la valeur de 180 min pour la constante intégrale τ_I est admise, et donc l'ajustement des paramètres du contrôleur est le suivant :

$K_c = -5{,}76$; $\tau_I = 180$ minutes ; $\tau_D = 0$

L'ajustement du contrôleur a été effectué sous les conditions d'opération suivantes :

- Copeaux d'alimentation à 50 % de consistance

- Point de consigne du contrôleur est à 16,4 kg/t de solides dissous.

La figure 5.14 présente la réponse du procédé lors d'un changement du point de consigne du contrôleur de 16,4 kg/t à 18,4 kg/t :

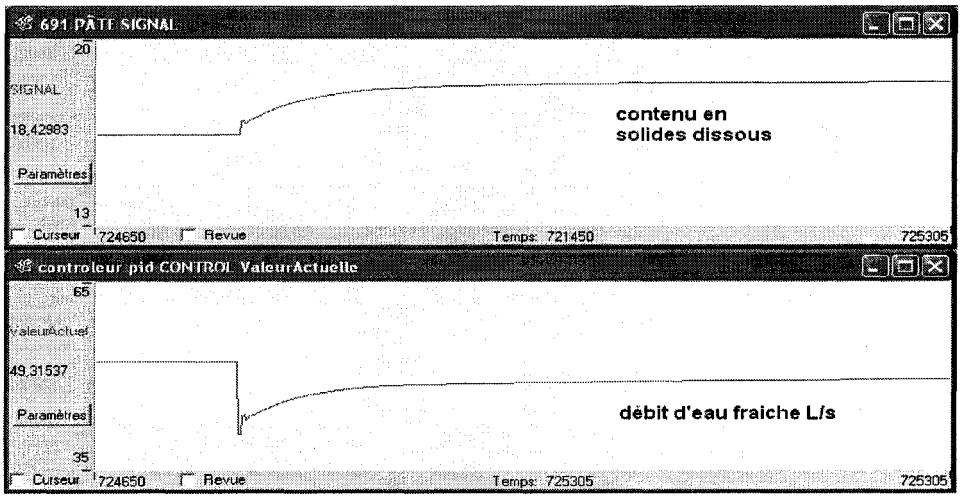


Figure 5.14 Réponse du procédé à un changement du point de consigne ($\lambda = 60$, 50% consistance)

L'ajustement avec la méthode Lambda permet d'atteindre le point de consigne avec une grande stabilité ce qui a pour avantage une diminution de la variabilité puisque la variable manipulé varie sans oscillation. Le choix de la constante de temps en boucle fermée λ est très important pour un contrôle stable et efficace pour la plage d'opération de 45% à 55 % de la consistance des copeaux. Une valeur plus faible de Lambda permet un contrôle plus agressif mais risque d'être inadéquate pour une région de la plage d'opération comme le montrent les figures 5.15 et 5.16 où les copeaux d'alimentation sont à 45 % et 50 % de consistance et Lambda vaut 50 min au lieu de 60 min. On constate une oscillation au niveau du débit d'eau lorsque les copeaux sont à 45% de consistance, ce qui n'est pas souhaitable pour un contrôle stable du procédé et de même les objectifs de l'ajustement par la méthode Lambda. Finalement la valeur de 60 min pour Lambda est choisie pour un contrôle plus robuste sur la plage d'opération.

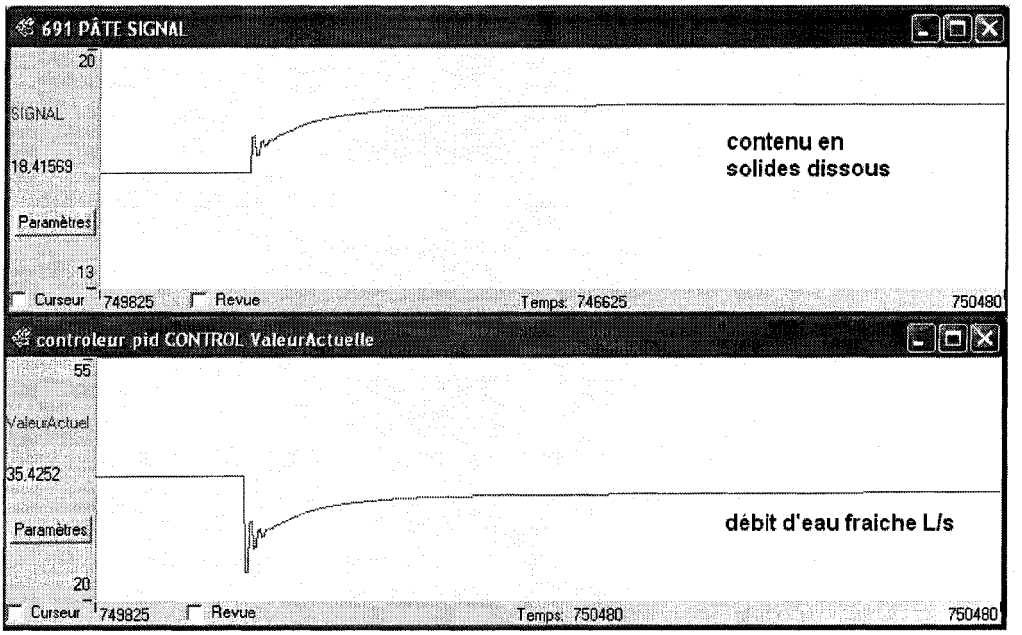


Figure 5.15 Réponse du procédé à un changement du point de consigne de 16,4 à 18,4 ($\lambda = 50$, 45% consistance)

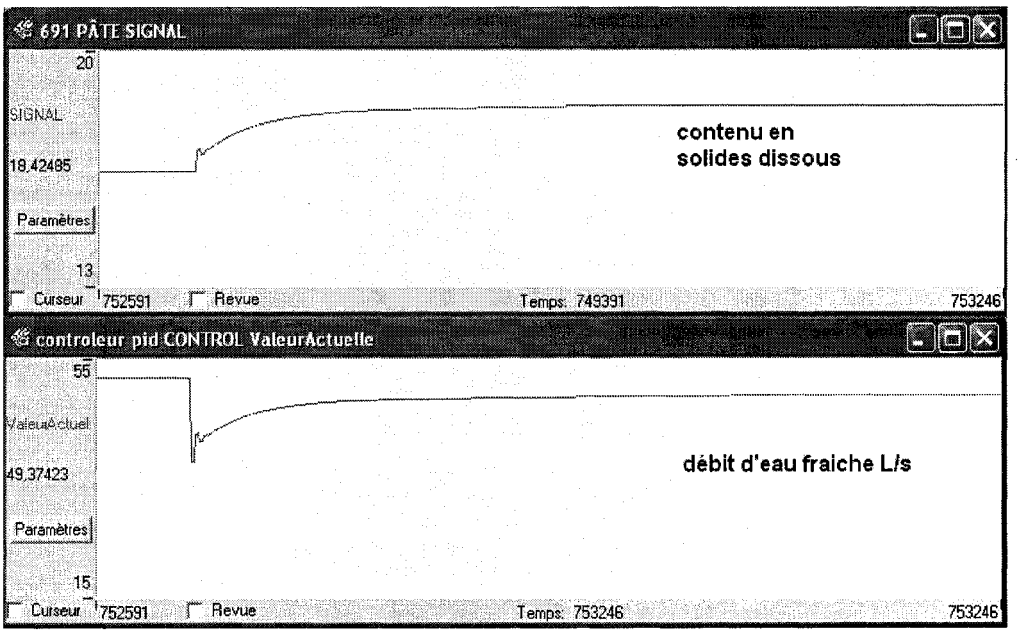


Figure 5.16 Réponse du procédé à un changement du point de consigne de 16,4 à 18,4 ($\lambda = 50$, 50% consistance)

L'inconvénient majeur de la méthode Lambda est percevable lorsque le procédé est perturbé. Une grande constante de temps affecte la réponse du contrôleur qui ne réagit

pas assez rapidement aux perturbations, ce qui équivaut à des coûts plus élevé et baisse de performance.

5.4.3.4 Programmation des gains

Le lavage de la pâte comporte plusieurs perturbations qui affectent le contrôle des solides dissous de la pâte lavée, la consistance est le facteur étudié dans cette simulation. Pour un meilleur contrôle, il faut introduire au contrôleur les paramètres optimaux qui dépendent où on se trouve sur la plage d'opération. La programmation des gains offre une bonne option. Le calcul des gains de procédé K_p peut se faire avec le module Jacobien pour chaque valeur de la consistance à l'alimentation (tableau 5.4).

Tableau 5.4 Gains de procédé optimaux Kp

Consistance (%)	Gain optimal K_p
45	-1,2
46	-0,994
47	-0,952
48	-0,925
49	-0,898
50	-0,867
51	-0,836
52	-0,806
53	-0,776
54	-0,745
55	-0,636

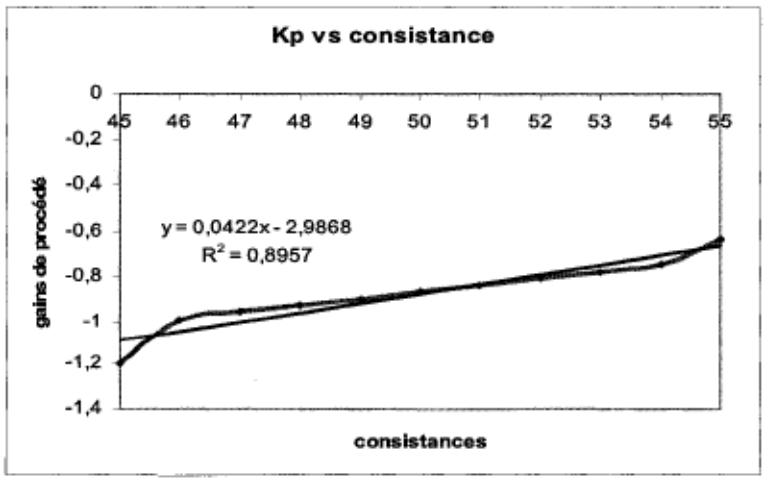


Figure 5.17 Changement du gain vs la consistance

Le paramètre Lambda est aussi déterminant pour augmenter la vitesse de réaction du contrôleur comme le démontre l'équation du calcul du gain K_c. Dans la simulation de lavage de pâte, le contrôle se fait avec un point de consigne optimal, on a alors plus besoin de réagir aux perturbations que de changer le point de consigne. La stratégie de programmation des gains sera axée sur l'introduction de gains plus élevés lorsque l'erreur entre la variable mesurée et la consigne est élevée et de gains plus faibles dans le cas contraire en utilisant les gains de procédé calculés pour la plage d'opération et en variant le paramètre Lambda dans le temps en fonction de l'erreur, s'inspirant ainsi du contrôleur à erreur carrée «error-squared controller» [6] :

$$K_c = K_{c0}\left(1 + a\left|e(t)\right|\right) \qquad \text{Éq. 5.13}$$

$$K_{c0} = \frac{\tau}{K_p \lambda_0} \qquad \text{Éq. 5.14}$$

$$K_c = \frac{\tau}{K_p \lambda} \qquad \text{Éq. 5.15}$$

Le gain K_p ne dépendant que de la consistance d'entrée des copeaux, et la constante de temps du contrôleur est gardée constante à 180 minutes. On peut déduire l'équation suivante :

$$\frac{1}{\lambda} = \frac{1}{\lambda_0}\left(1 + a\left|e(t)\right|\right) \qquad \text{Éq. 5.16}$$

λ_0 correspond à la valeur de Lambda lorsque l'erreur est nulle (λ_0 = 60 minutes), et le paramètre a est réglé par l'utilisateur pour augmenter l'agressivité du contrôleur.

La stratégie de programmation des gains est introduite dans la simulation par une logique qui permet de changer le gain du procédé selon la consistance en s'appuyant sur l'équation de régression linéaire du graphique de la figure 5.17 et la variation du paramètre Lambda en fonction de l'erreur (figure 5.18).

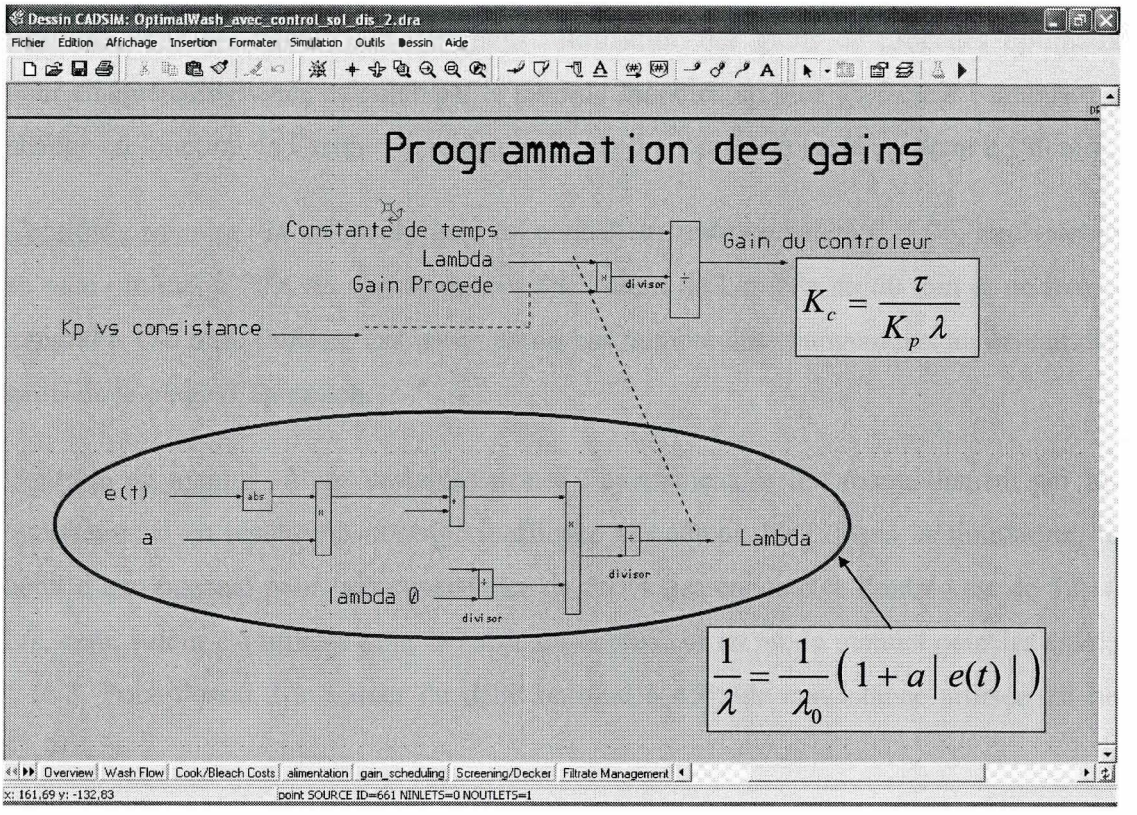


Figure 5.18 Logique permettant de changer le gain du contrôleur.

5.5 Résultats

5.5.1 Réduction des coûts de production et de la variabilité

Dans la simulation de contrôle de lavage, des essais sont effectués pour évaluer la performance du contrôleur suite à trois perturbations au niveau de la consistance des copeaux à l'entrée. La variation au niveau de la consistance est un problème important auquel sont confrontés les usines. Elle peut atteindre 20 % (60%-40% de consistance) dans une seule journée [33]. La performance de trois types de contrôle est évaluée au niveau de la réduction des coûts et la diminution de la variabilité :

Contrôle avec programmation des gains : utilisation des gains de procédés optimaux pour chaque consistance calculée par le module Jacobien en plus du facteur Lambda qui change en fonction de l'erreur, le temps intégrale du contrôleur reste constant à 180 min.

Contrôle ordinaire : la valeur du gain du contrôleur reste inchangé à -5,76, équivalente au gain optimal à 50% de consistance et une valeur de Lambda de 60 min et un temps intégral à 180 min. Dans ce cas on ne prend pas compte des changements au niveau des gains de la plage d'opération.

Contrôle en manuel : Dans certaines usines l'ajustement se fait manuellement par les opérateurs, il en résulte un contrôle inefficace et coûteux [34]. Dans la simulation, le débit d'eau optimal pour une consistance de 50 % des copeaux à l'entrée est de 52,16 L/s, cette valeur est utilisée pour la sortie du contrôleur en mode manuel pour les essais 1 et 3. Pour l'essai 2 la valeur du débit optimal à 45% de consistance utilisée est de 37,91 L/s.

5.5.2 Essai n°1

Le premier essai permet des perturbations de la consistance des copeaux dans la plage d'opération de 45 % à 55% pour une journée de production (1440 minutes), le tableau 5.5 et la figure 5.21 expliquent le déroulement de la perturbation de type rectangulaire.

Tableau 5.5 Signal de perturbation de la consistance (essai 1)

Changement de la consistance (%)	50 à 45	45 à 55	55 à 45
Durée (minutes)	240	600	600

5.5.2.1 Réduction des coûts d'opération

Tableau 5.6 Coûts de production selon le paramètre «a».

paramètre a	coût de production ($)	coût ($/t)
0	11450	12,87
1	11414	12,83
2	11403	12,81
3	11401	12,81
4	11399	12,81

Le tableau 5.6 montre que les coûts de production (coûts d'évaporation, produits chimiques de cuisson et de blanchiment) sont presque égaux pour a = 2,3,4. La valeur de a = 3 est la valeur maximale qui peut être utilisée sans risque d'oscillation de la variable manipulée comme le montre la figure 5.19 où une valeur de «a» égale à 4 a été utilisée.

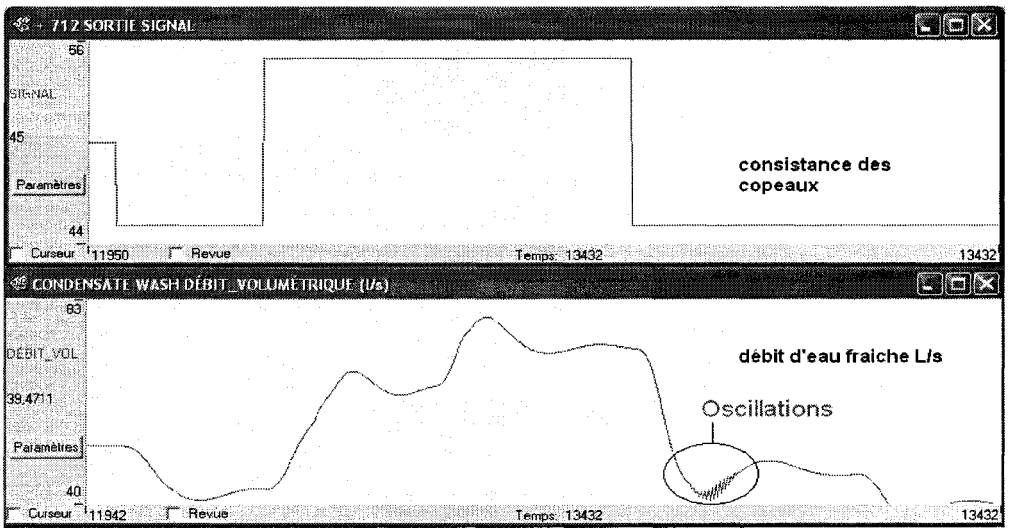


Figure 5.19 Oscillation de la sortie du contrôleur en réponse à la perturbation pour a =4.

Une valeur nulle du paramètre «a» équivaut à un contrôle avec seulement un changement au niveau des gains du procédé et un Lambda fixe à 60 minutes. La valeur «a = 3» sera utilisé pour comparer la performance du contrôle par programmation des gains avec les autres types de contrôle.

Tableau 5.7 Coûts d'opérations et consommation en eau de l'essai 1

Type de contrôle	Coût d'opération($/j)	Coût ($/t)	Quantité d'eau moyenne (L/s)	Quantité d'eau totale (t/j)
Contrôle ordinaire	**11457**	**12,87**	**54,72**	**4731,5**
Mode manuel	**11685**	**13,13**	**52,16**	**4506,6**
Programmation des gains	**11401**	**12,81**	**54,49**	**4711,3**

Par rapport au contrôle ordinaire la programmation des gains (a =3) permet une réduction de 20440$/ année (56$/j) des coûts de production. La réduction est plus importante par rapport au contrôle manuel, elle atteint alors 103660 $/année (284$/j) d'économie possible.

La programmation des gains avec (a =0) ne procure pas une grande réduction des coûts de productions par rapport au contrôle ordinaire avec seulement un gain de (7$/j) car le gain du procédé ne change pas beaucoup dans la fourchette de 45% à 55% de consistance comme l'indique le tableau 5.6.

La quantité d'eau utilisée en mode manuel est plus faible ce qui a pour effet une plus grande perte en produits chimiques et organiques (figure 5.20). La programmation des gains utilise 20 tonnes d'eau de moins que le contrôle ordinaire en plus de réduire le coût et la variabilité.

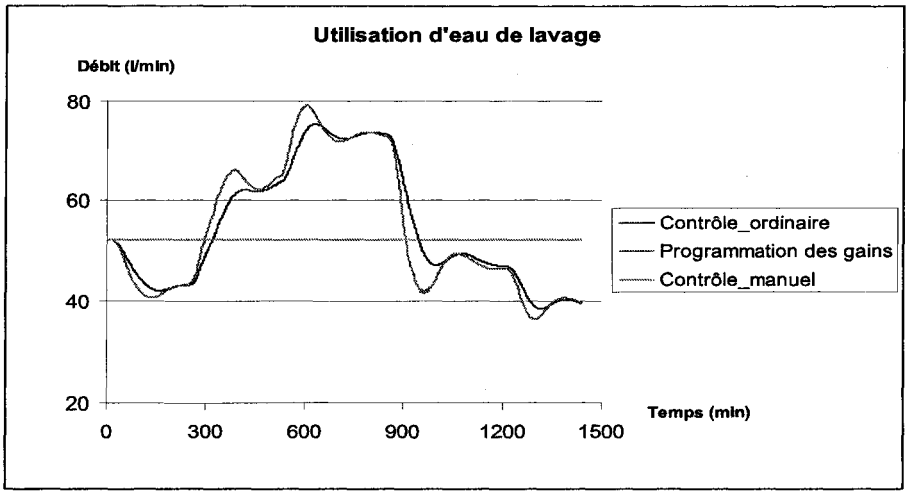


Figure 5.20 Utilisation d'eau de lavage (essai n°1).

5.5.2.2 Réduction de la variabilité

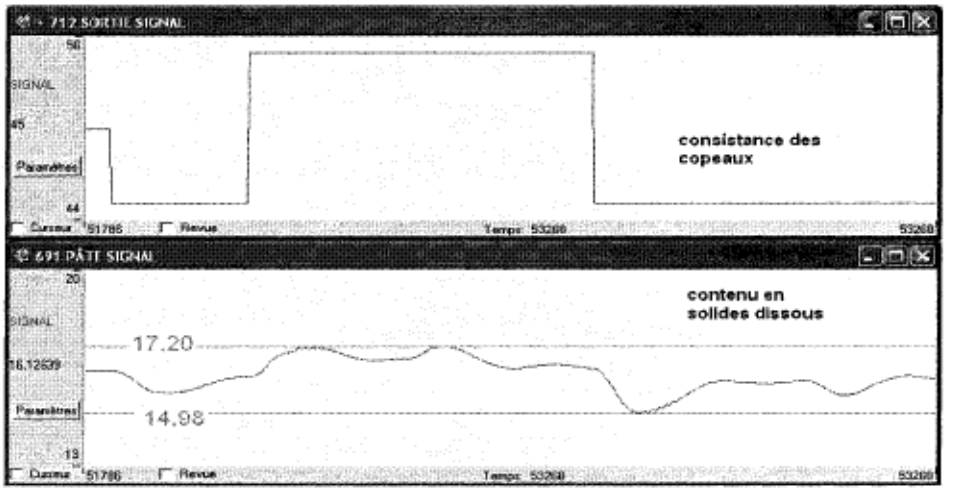
Le calcul de la variabilité est une mesure de l'étendue de la variation d'une variable donnée en utilisant la moyenne et l'écart type.

$$Variabilité\,(\%) = 100\,\frac{2\sigma}{Moyenne} \qquad \text{Éq. 5.17}$$

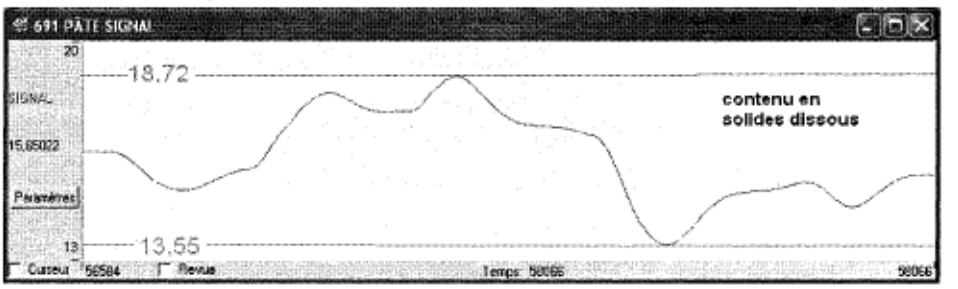
L'écart type σ est une mesure de la dispersion des mesures. Lorsque la distribution statistique est normale, le double de l'écart type «2σ» présente un intervalle de confiance de 95%. La variabilité est exprimée en terme de pourcentage sans unité pour permettre la comparaison des variables ayant des unités différentes [1].

Les résultats du tableau 5.8 et la figure 5.21 montrent que le contrôle des solides dissous par la programmation des gains a permis une meilleure réduction de la variabilité (7,04 %) ce qui a pour effet de maîtriser la variable contrôlée dans un intervalle étroit et d'être la plus proche possible du point de consigne optimal avec un maximum à 17,26 kg/t et un minimum à 14,89 kg/t. On constate une plus grande variabilité au niveau du contrôle manuel, une perte significative en solides dissous (jusqu'à 30 kg/t) ceci peut être

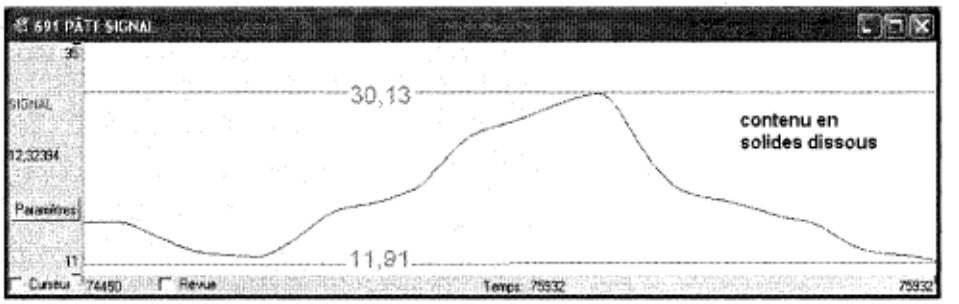
expliqué par l'utilisation d'une quantité insuffisante d'eau de lavage, ce qui implique des coûts plus élevés des produits chimiques.



(a)



(b)



(c)

Figure 5.21 **Variabilité du contenu en solides dissous de la pâte. (a) programmation des gains, (b) contrôle ordinaire, (c) contrôle en mode manuel.**

Tableau 5.8 Variabilité des solides dissous pour l'essai 1

Type de contrôle	Moyenne	Écart type	Variabilité	Maximum	Minimum
Contrôle ordinaire	16,22 kg/t	1,39 kg/t	17,08%	18,72 kg/t	13,55 kg/t
Mode manuel	18,84 kg/t	5,55 kg/t	58,90%	30,13 kg/t	11,91 kg/t
Programmation des gains	16,24 kg/t	0,57 kg/t	7,04%	17,20 kg/t	14,98 kg/t

5.5.3 Essai n°2

Le deuxième essai de la perturbation est aussi un signal rectangulaire, avec des changements brusque au niveau de la consistance de la limite inférieur 45% à la limite supérieure de 55% de la plage d'opération pour une journée comme le montre le tableau 5.9 et la figure 5.23.

Tableau 5.9 Signal de perturbation de la consistance (essai 2)

Changement de la consistance (%)	45 à 55	55 à 45
Durée (minutes)	720	720

5.5.3.1 Réduction des coûts d'opération

Tableau 5.10 Coûts d'opérations et consommation en eau de l'essai 2

Type de contrôle	Coût d'opération($/j)	Coût ($/t)	Quantité d'eau moyenne (L/s)	Quantité d'eau totale (t/j)
Contrôle ordinaire	11646	12,89	54,93	4749,4
Mode manuel	13258	14,63	37,91	3275,4
Programmation des gains	11581	12,78	54,91	4747,4

La programmation des gains a permis une réduction de 23725$/ année (65$/j) des coûts de la production par rapport au contrôle ordinaire. La réduction est encore plus importante par rapport au contrôle manuel, soit 612 105 $/année (1677$/j). Cela est dû à la grande perturbation de 10% de la consistance avec une quantité d'eau de lavage insuffisante de 37,91 l/s , ce qui produit une grosse pertes des solides dissous. En réalité, l'opérateur devrait faire les ajustements nécessaires pour compenser l'effet de la perturbation.

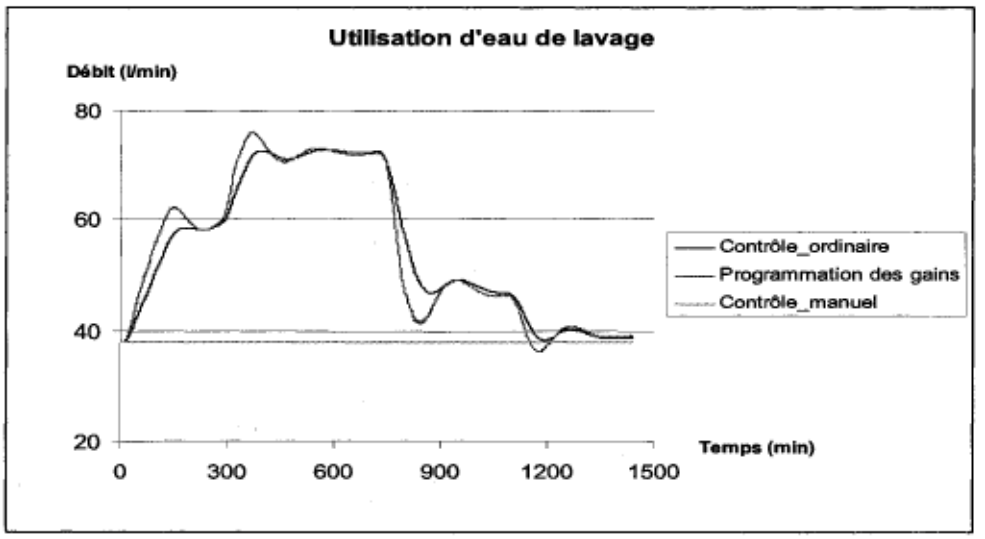


Figure 5.22 Utilisation d'eau de lavage (essaie n°2).

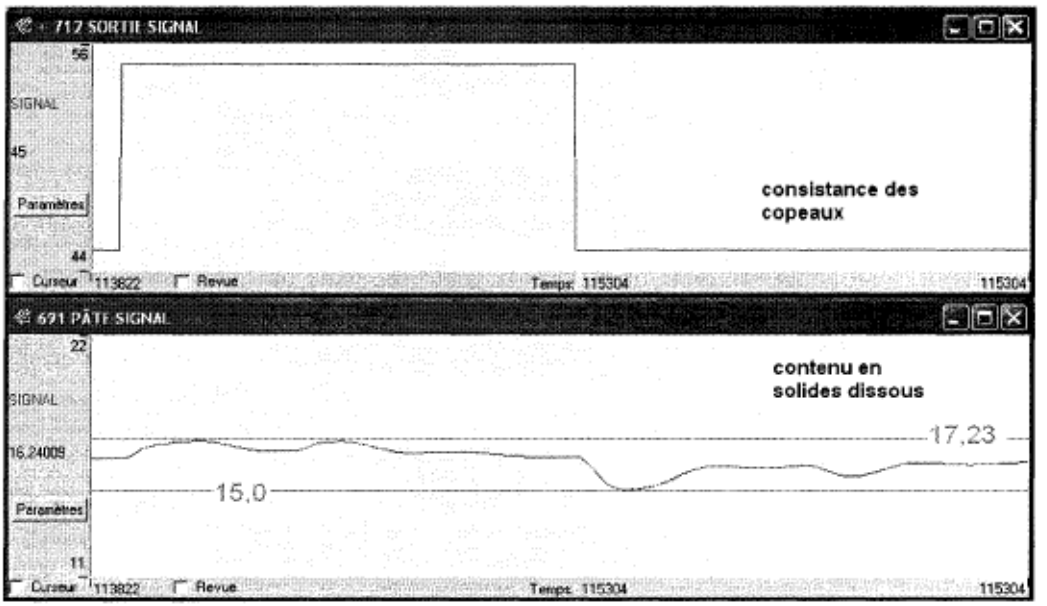
En mode manuel la quantité d'eau de lavage utilisée est optimale pour une consistance d'alimentation de 45%. L'utilisation d'eau fraîche de lavage est similaire pour le contrôle ordinaire et la programmation des gains, pourtant ce dernier permet une meilleure réduction des coûts et de la variabilité, ceci peut être expliqué par une utilisation efficace de l'eau de lavage comme le montre la figure 5.22 où une plus grande quantité d'eau est utilisée dans les premiers 8 heures et une plus faible quantité dans les derniers 6 heures pour essayer de maintenir la variable contrôlée le plus près possible du point de consigne optimal.

5.5.3.2 Réduction de la variabilité

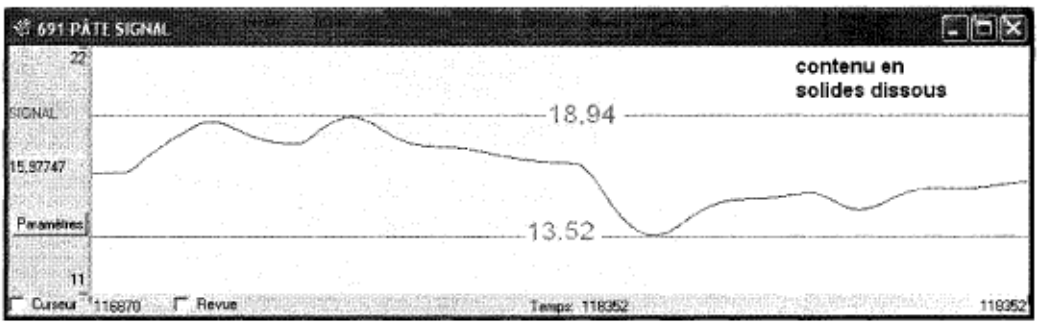
Le contrôle par programmation des gains a encore démontré de bons résultats pour maintenir les solides dissous avec une faible variabilité (figure 5.23). La teneur moyenne en solides dissous par contrôle ordinaire est semblable à celle de la programmation des gains, mais la variabilité est 2 fois plus importante, la gestion d'eau de lavage en est responsable. Le contrôle manuel montre une très grande variabilité avec un gaspillage énorme des produits chimiques de cuisson et des organiques (60,79 kg/t) qui ne peuvent être récupérés, en plus de l'impact sur la qualité de la pâte dans les stades subséquent (tableau 5.11).

Tableau 5.11 Variabilité des solides dissous pour l'essai 2

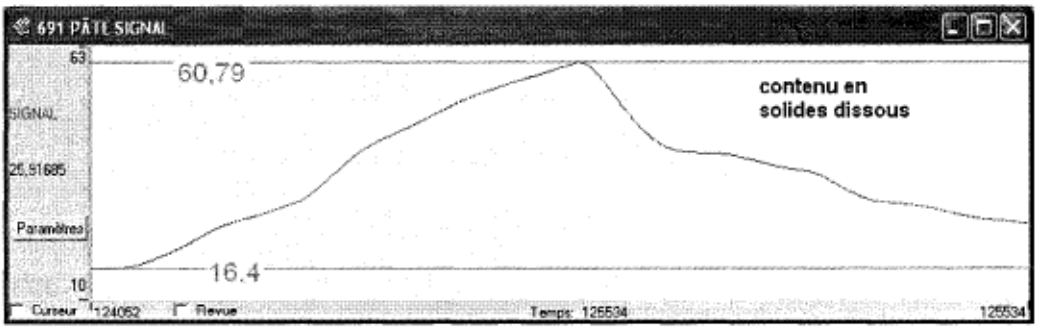
Type de contrôle	Moyenne	Écart type	Variabilité	Maximum	Minimum
Contrôle ordinaire	16,47 kg/t	1,50 kg/t	18,22 %	18,94 kg/t	13,52 kg/t
Mode manuel	38,18 kg/t	12,07 kg/t	63,23 %	60,79 kg/t	16,40 kg/t
Programmation des gains	16,34 kg/t	0,57 kg/t	6,99 %	17,23 kg/t	15 kg/t



(a)



(b)



(c)

Figure 5.23 **Variabilité du contenu en solides dissous de la pâte. (a) programmation des gains (a=3), (b) contrôle ordinaire, (c) contrôle en mode manuel.**

5.5.4 Essai n°3

Le troisième essai est effectué selon une perturbation de type sinusoïdale avec une période de 320 minutes équivalente à un changement de 5% de consistance chaque 80 minutes et cela pendant une journée (figure 5.25).

5.5.4.1 Réduction des coûts d'opération

Tableau 5.12 Coûts d'opérations et consommation en eau de l'essai 3

Type de contrôle	Coût d'opération($/j)	Coût ($/t)	Quantité d'eau moyenne (l/s)	Quantité d'eau totale (t/j)
Contrôle ordinaire	**10015**	**11,65**	**60,39**	**5221,7**
Mode manuel	**10638**	**12,38**	**52,16**	**4506**
Programmation des gains	**9848**	**11,46**	**60,27**	**5210,6**

La programmation des gains a permis une réduction de 60955$/ année (167$/j) des coûts de la production par rapport au contrôle ordinaire. La réduction est très importante par rapport au contrôle manuel, soit 288350 $/année (790$/j).

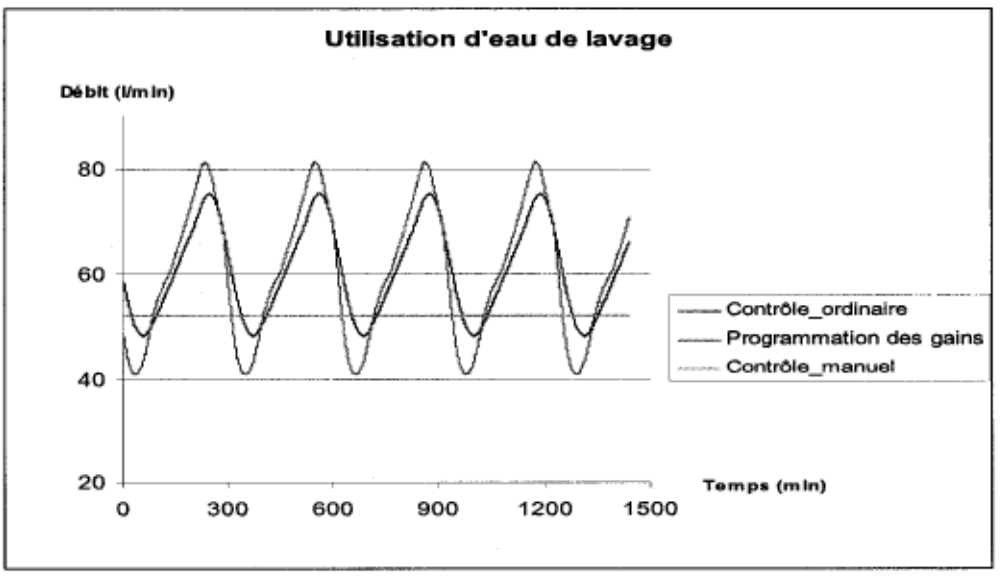


Figure 5.24 Utilisation d'eau de lavage (essai n°3).

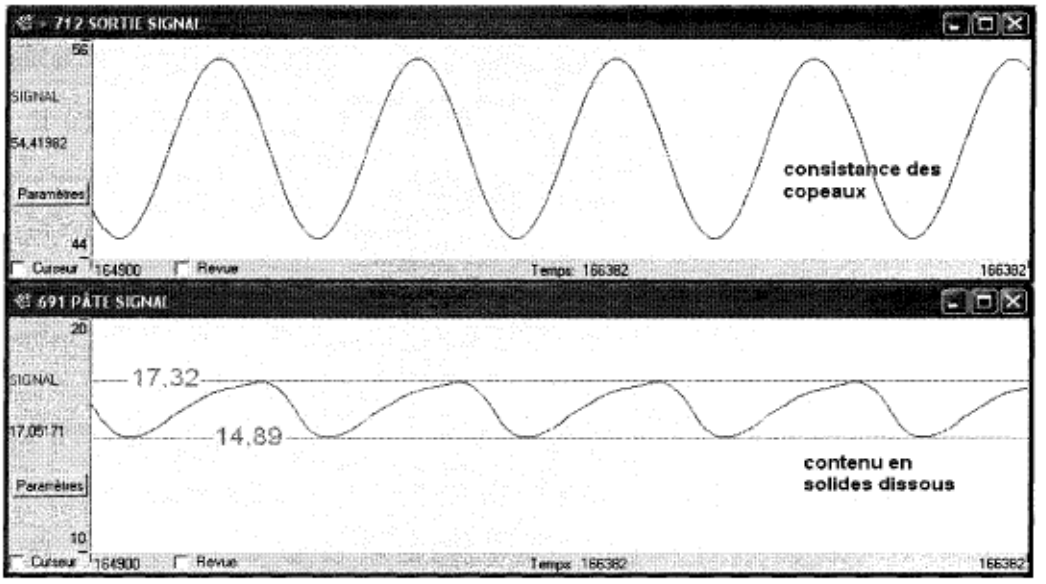
La bonne gestion d'eau fraîche de lavage par le contrôle avec programmation des gains a permis de réduire les coûts de production en plus de la variabilité (figure 5.24).

5.5.4.2 Réduction de la variabilité

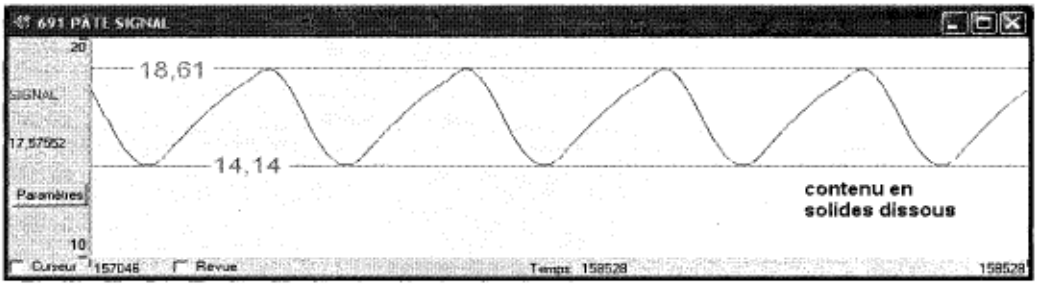
Le contrôle par programmation des gains se distingue avec un meilleur contrôle de la variabilité (10 %) des solides dissous. La moyenne en contenu des solides dissous du contrôle ordinaire est très proche de celle par programmation des gains mais avec une variabilité 2 fois plus élevée (tableau 5.13). Le contrôle manuel n'est pas du tout optimal avec des contenus en solides dissous très supérieurs au point de consigne optimal, en plus d'une plus grande variabilité (figure 5.25).

Tableau 5.13 Variabilité des solides dissous pour l'essai 3

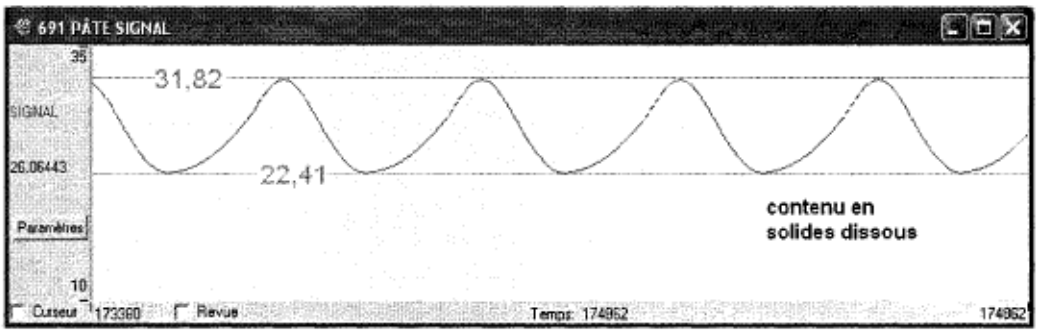
Type de contrôle	Moyenne	Écart type	Variabilité	Maximum	Minimum
Contrôle ordinaire	16,28 kg/t	1,48 kg/t	18,13 %	18,61 kg/t	14,14 kg/t
Mode manuel	26,05 kg/t	3,18 kg/t	24,40 %	31,82 kg/t	22,41 kg/t
Programmation des gains	16,16 kg/t	0,84 kg/t	10,40 %	17,32 kg/t	14,89 kg/t



(a)



(b)



(c)

Figure 5.25 **Variabilité du contenu en solides dissous de la pâte. (a) programmation des gains (a=3), (b) contrôle ordinaire, (c) contrôle manuel.**

5.6 Conclusions

À la lumière des résultas obtenus pour différents signaux, l'ampleur de la réduction des coûts d'opération et de la variabilité est liée au type de signal de perturbation de la consistance des copeaux. Le contrôle par programmation des gains qui utilise des gains optimaux de procédé à partir du module Jacobien offre un meilleur contrôle des solides dissous avec un coût plus faible que le contrôle ordinaire et une meilleur gestion d'eau de lavage. Le contrôle manuel utilisé dans certaines usines, augmente les coûts de production et de la variabilité avec des pertes de produits chimiques.

Les réductions au niveau des coûts d'opération seraient encore plus importantes si on considérait les coûts liés au chauffage de l'eau de lavage, aux pertes des organiques servant à la combustion, au traitement des effluents et au pompage.

Chapitre 6 - Contrôle multivariable d'un mélangeur

La grande majorité des procédés comportent plusieurs variables manipulées et contrôlées, l'ajustement des contrôleurs doit prendre en considération les effets d'interaction des boucles qui ont un impact direct sur la stabilité la performance du contrôle. La mesure de l'interaction par la MGR est très utile pour choisir la meilleure paire de variables manipulée-contrôlée qui minimise les interactions et l'ajustement du contrôleur.

6.1 Description de la simulation

La simulation de la figure 6.1 présente un mélangeur où deux débits sont manipulés, soit un liquide dilué (2% consistance) et l'autre concentré (70% consistance) pour contrôler la consistance et le débit de la sortie.

Les variables manipulée et contrôlée doivent être définies dans le module du Jacobien pour calculer les gains en boucle ouverte (Jacobien dy_i/dx_i) et les valeurs de la MGR .

Les variables «x_1, x_2, y_1, y_2» sont définit comme suit :

x_1 : Variable manipulée du flux diluée

x_2 : Variable manipulée du flux concentrée

y_1 : Variable contrôlée du débit total à la sortie du mélangeur

y_2 : Variable contrôlée de la consistance à la sortie du mélangeur

Deux contrôleurs de type PI sont utilisés pour atteindre les objectifs, le contrôleur de débit agit sur le débit du liquide dilué et le contrôleur de consistance ajuste le débit du liquide concentré.

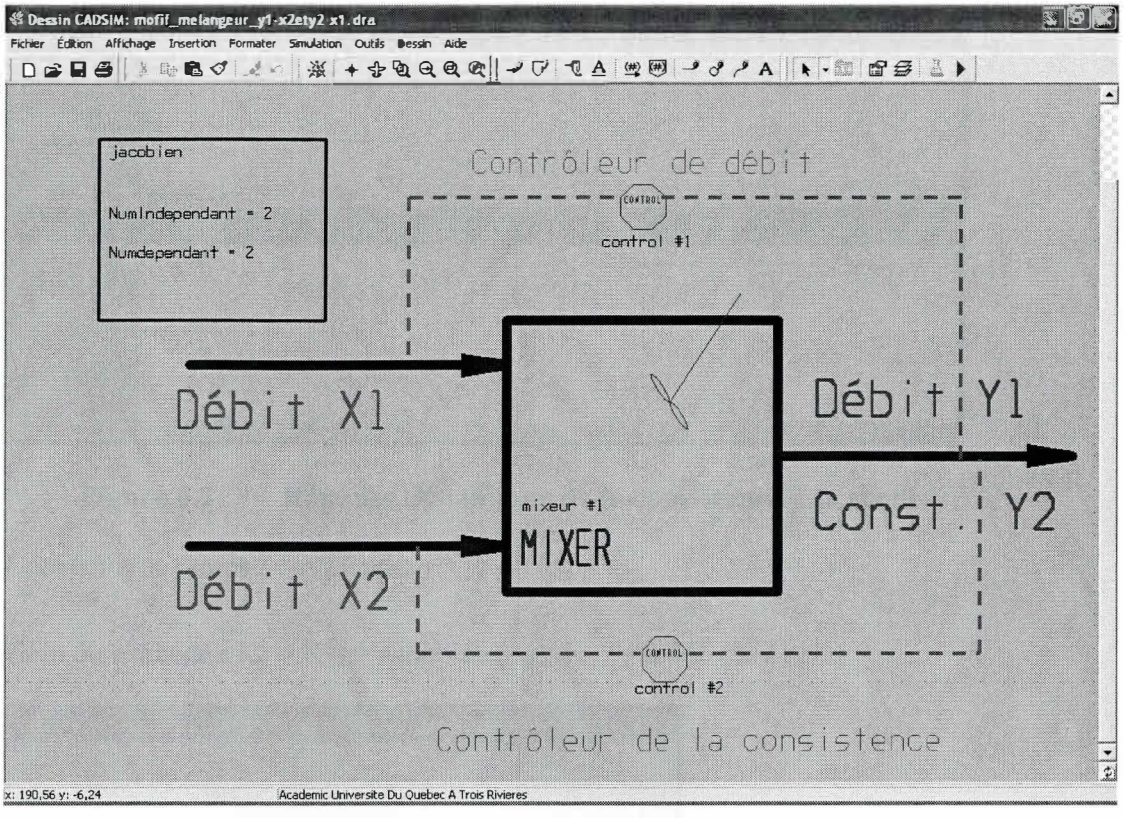


Figure 6.1 Contrôle multi-variable d'un mélangeur.

6.2 Ajustement des contrôleurs

6.2.1 Contrôleur de consistance

La figure 6.2 montre la réponse de la consistance y_2 suite à une perturbation de 10% du débit concentré de la variable x_2 (170,6 kg/s à 187.7 kg/s). La réponse est du premier ordre, on peut déduire la valeur de la constante de temps en plus du gain du procédé.

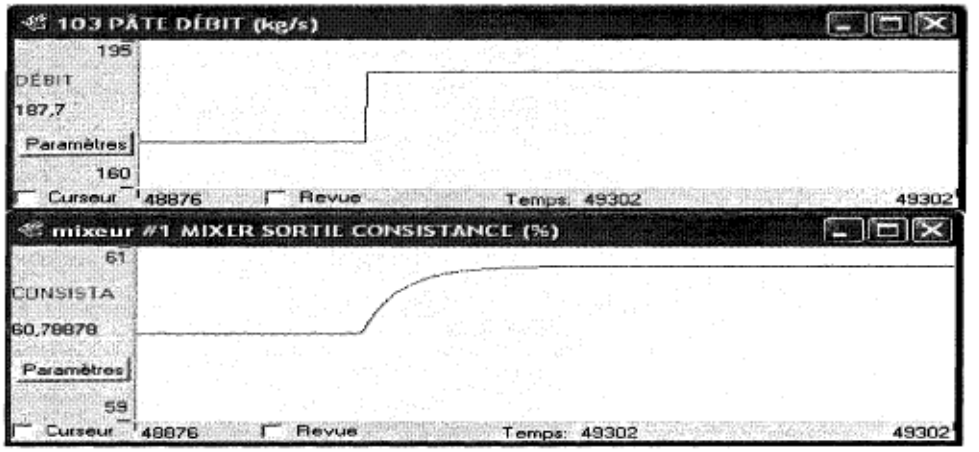


Figure 6.2 Réponse dynamique de la consistance à la sortie.

Constante de temps : $\tau = 21$ min

Gain du procédé : $K_p = K_{22} =$ Jacobien $(dy_2/dx_2) = 0.046$ (% / kg/s)

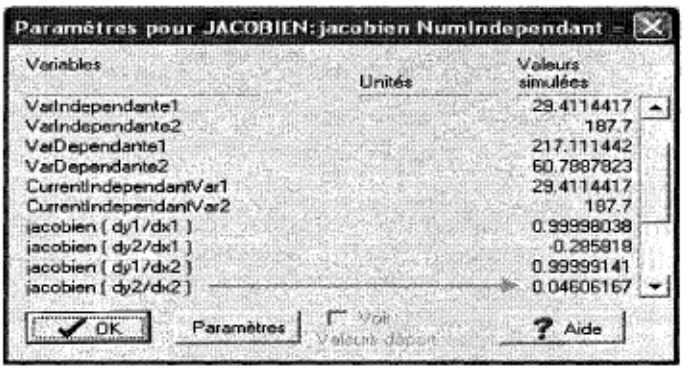


Figure 6.3 Gain du procédé à partir du module Jacobien.

Le facteur λ est choisi de façon à avoir une bonne réponse du contrôleur suite à un changement de point de consigne ou une perturbation. Après des essais de différentes valeurs, la valeur retenue de Lambda est de 5 minutes. Les paramètres du contrôleur PI sont :

$$K_c = \frac{\tau}{K_p \lambda} = 91{,}3 \text{ (kg/s / \%)}$$

$$\tau_I = \tau = 21 \text{ minutes}$$

6.2.2 Contrôleur de débit

La boucle de débit est très rapide par rapport à celle de la consistance avec une constante de temps de 1 minute et un gain de procédé K_{11} de 1. La valeur de Lambda choisie est de 3 minutes. La même procédure que dans l'ajustement du contrôleur de la consistance est utilisée pour trouver le gain et le temps intégral du contrôleur :

$K_c = 0,33$

$\tau_I =$ 1 minute.

6.3 Effet de l'interaction des boucles

6.3.1 Gain du procédé

On a vu dans la partie théorique que l'effet d'interaction des boucles survient lorsqu'une variable manipulée affecte plus d'une variable contrôlée. On peut d'ailleurs constater ces interactions dans la figure 6.3 en examinant les valeurs du Jacobien dy_2/dx_1 et dy_1/dx_2. La figure 6.4 montre le changement au niveau du gain de la boucle de la consistance lorsque la boucle de contrôle de débit est ouverte, et ensuite lorsqu'elle est fermée.

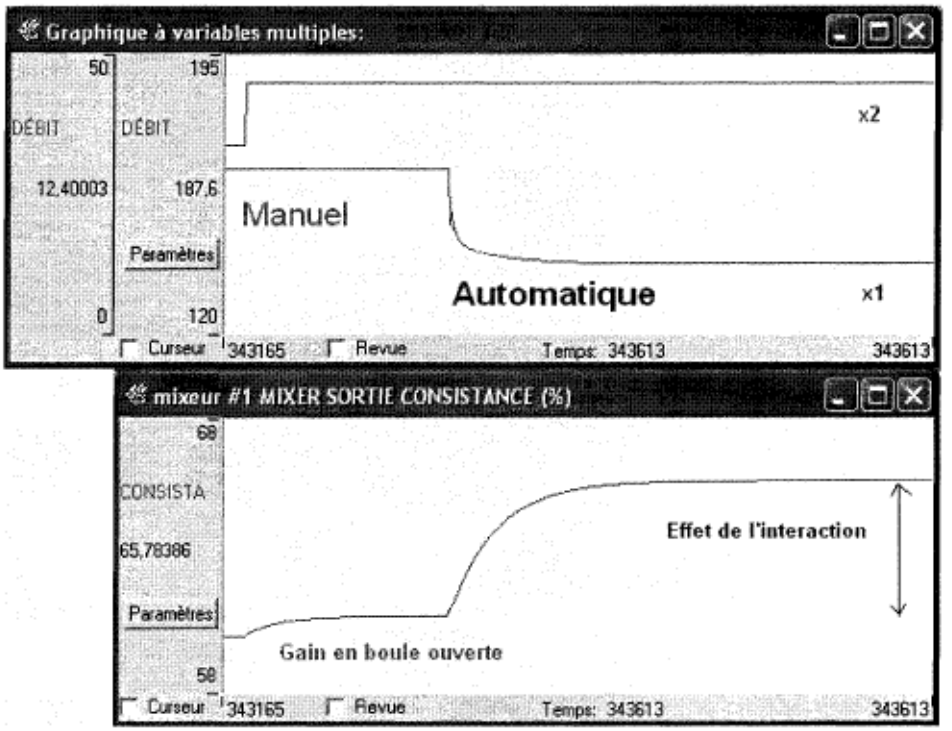


Figure 6.4 Effet de l'interaction sur le gain du procédé.

Lorsque les deux boucles sont ouvertes, le gain au niveau de la consistance y_2 lors d'une perturbation de 10 % du débit x_2 du liquide concentré est :

$$K_{22} = \frac{60,78 - 60}{187,6 - 170,6} = 0,046\ (\%/kg/s)$$

Le gain en boucle fermée K'_{22} est calculé lorsque la boucle de débit x1 est en automatique :

$$K'_{22} = \frac{65,78 - 60}{187,6 - 170,6} = 0,34\ (\%/kg/s)$$

On constate une forte interaction avec un gain en boucle fermée sept fois plus important que le gain en boucle ouverte, le gain relatif peut être calculé par le ratio du gain en boucle ouverte par rapport au gain en boucle fermée :

$$\lambda_{22} = \frac{K_{22}}{K'_{22}} = 0,135$$

6.3.2 Mesure de l'interaction par le module Jacobien

Bristol a proposé une méthode pour mesurer l'interaction des boucles en calculant la MGR. Le module Jacobien permet de calculer la matrice des gains K en boucles ouverte et ensuite calculer la MGR en effectuant des calculs matriciels selon l'équation 2.11:

$$MGR = K \otimes (K^{-1})^T$$

En effectuant les perturbations nécessaires, le module Jacobien donne la MGR suivante :

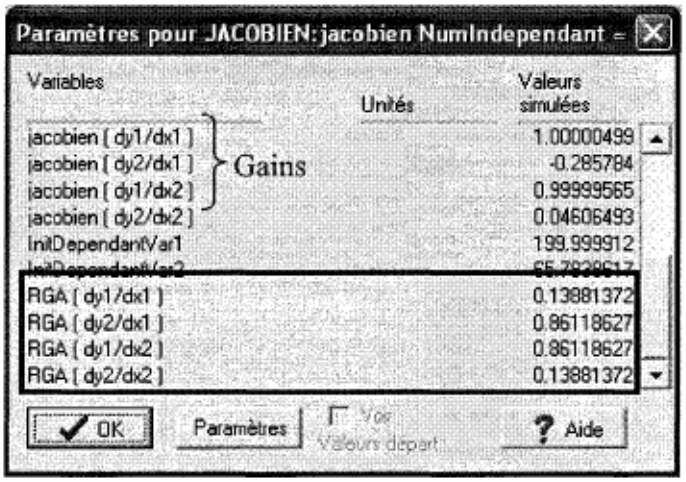


Figure 6.5 MGR.

$$\lambda_{ij} = \begin{matrix} \\ y_1 \\ y_2 \end{matrix} \begin{matrix} \begin{matrix} X1 & X2 \end{matrix} \\ \begin{pmatrix} 0{,}1388 & 0{,}8612 \\ 0{,}612 & 0{,}1388 \end{pmatrix} \end{matrix}$$

Les résultats de la MGR confirment deux propriétés du MGR :

- Pour un système avec deux variables manipulées et deux contrôlées, $\lambda_{22} = \lambda_{11} =$ 0,1388 et $\lambda_{12} = \lambda_{21} = 0{,}8612$.
- La somme des lignes est égale à l'unité et de même pour les colonnes.

La MGR (figure 6.5) nous informe que les paires y_1-x_2 et y_2-x_1 sont souhaitables pour un meilleur contrôle qui minimise l'interaction puisque λ_{12} et λ_{21} (0,8612) sont proches de

1, et donc le gain n'augmente que par un facteur de 1.16 lorsque la boucle est fermée. Par contre, les paires y_1-x_1 et y_2-x_2 comportent beaucoup plus d'interaction avec un gain relatif de 0,1388, ce qui signifie que le gain en boucle fermée est 7.2 fois plus important que le gain en boucle ouverte, ces paires sont donc à éviter. Le gain relatif λ_{22} de la paire y_2-x_2 calculé par le Jacobien est égal à celui calculé manuellement.

6.3.3 Performance du contrôle

La figure 6.6 montre une bonne performance du contrôleur de la consistance lorsqu'on effectue un changement de point de consigne de 60% à 55% avec les paramètres du contrôleur PI calculés précédemment (K_c = 91,3 (kg/s / %), τ_I = 21 minutes) tout en maintenant ouverte la boucle de contrôle de débit. La variable contrôlée de la consistance y_2 atteint le point de consigne sans dépassement.

La performance est moins bonne lorsqu'on met le contrôleur de débit en mode automatique (figure 6.7). La variable manipulée x_2 change avec une grande variation ce qui provoque un large dépassement de la variable contrôlée y_1, ceci est engendré par l'effet de l'interaction des boucles. Ce large dépassement n'est pas désirable pour la performance du contrôle d'un procédé car il peut induire de l'instabilité et de la variabilité dans les stades subséquents. Le contrôleur doit être réajusté pour prendre compte de l'effet de l'interaction.

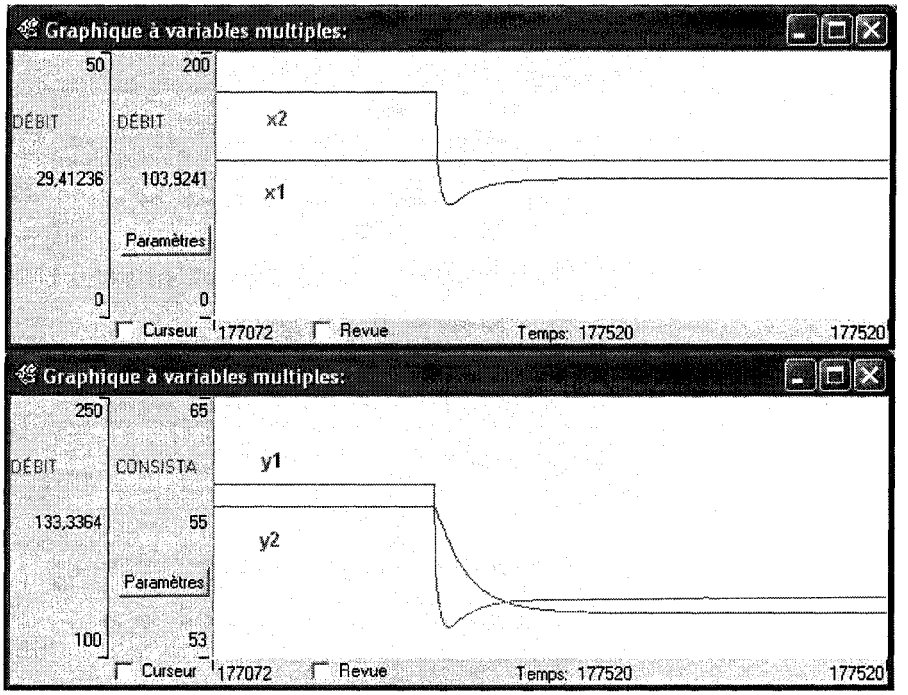


Figure 6.6 Changement du point de consigne avec x_1 en manuel .

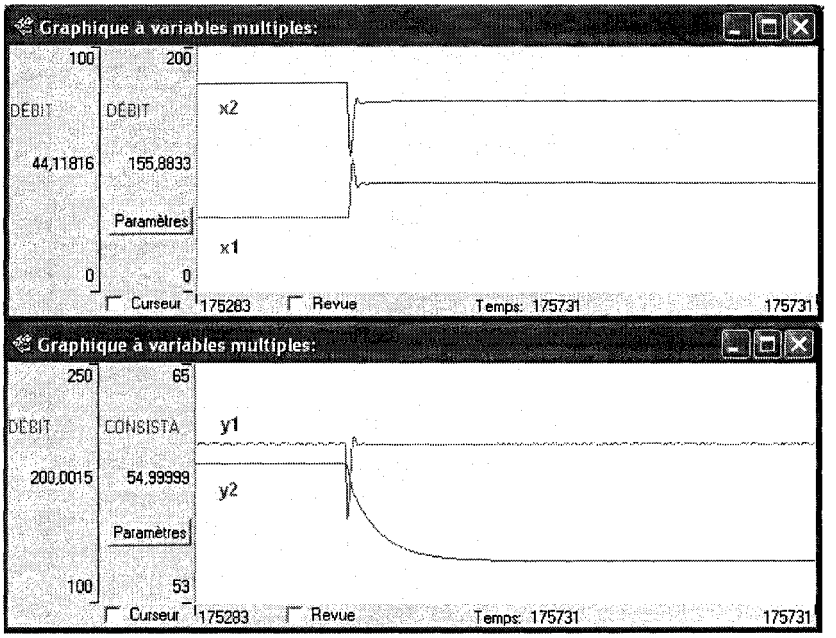


Figure 6.7 Changement du point de consigne avec x_1 en automatique.

6.4 Contrôle multivariable

La performance des contrôleurs est médiocre du fait que les paires y_1-x_1 et y_2-x_2 comportent beaucoup d'interactions avec un gain relatif de «0,138», pour un contrôle de consistance à 60%. Pour régler ce problème, la première étape est de choisir la meilleure paire de variables «manipulée-contrôlée» qui minimise l'interaction. Dans notre cas, le résultat du MGR du module Jacobien de la figure 6.8 démontre qu'on doit contrôler la consistance «y_2» avec le débit diluée «x_1» et manipuler le débit concentré «x_2» pour contrôler le débit de sortie du mélangeur «y_1».

Maintenant que les paires y_1-x_2 et y_2-x_1 sont choisies, l'ajustement des paramètres des contrôleurs peut se faire suivant une procédure développée par Shinskey qui consiste à identifier la boucle la plus rapide et l'ajuster avec la boucle lente en manuel, puis ajuster la boucle lente avec la boucle rapide en automatique pour tenir compte des interactions [7]. Dans l'exemple du mélangeur, la boucle la plus lente est celle qui contrôle la consistance avec une constante de temps de 21 minutes.

Avant d'ajuster les contrôleurs, on effectue une perturbation pour calculer les gains en boucle ouverte et les gains relatifs, la figure 6.8 montre le résultat:

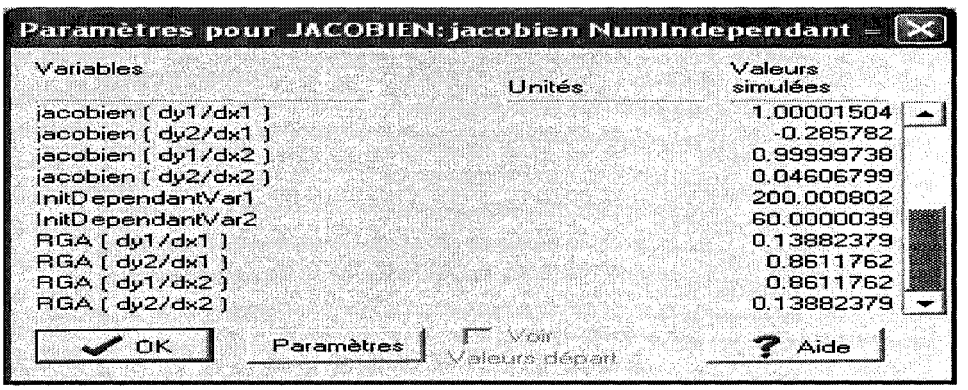


Figure 6.8 Calcul des gains en boucle ouverte Jacobien(dy/dx) et la MGR.

6.4.1 Ajustement du contrôleur de débit

La boucle de contrôle du débit est ajustée avec la boucle de consistance en mode manuel, les paramètres du contrôleur PI calculés sont :

$$K_c = \frac{\tau}{K_{12}\,\lambda}$$ avec : $\tau = 1$ minute , $\lambda = 3$ minutes , K_{12} = Jacobien $(dy_1/dx_2) = 1$.

$K_c = 0.33$; $\tau_I = 1$ minute.

6.4.2 Ajustement du contrôleur de consistance

Le contrôleur de débit est mis en mode automatique, les paramètres du contrôleur calculés avec la méthode Lambda sont :

$$K_c = \frac{\tau}{K'_{21}\,\lambda}$$ avec : $\tau = 21$ minute , $\lambda = 5$ minutes ,

$$K'_{21} = \frac{61-60}{26{,}6-29{,}41} = -0{,}34\ (\%\ /\ \text{kg/s}).$$

$K_c = -12{,}35$ (kg/s / %) , $\tau_I = 21$ minutes .

On peut aussi calculer le gain K'_{21} en boucle fermée du contrôleur de consistance à partir du gain en boucle ouverte K_{21} et le gain relatif calculés par le module Jacobien (figure 4.41) :

$$K'_{21} = \frac{K_{21}}{\lambda_{21}} = \frac{jacobien(dy_2/dx_1)}{RGA(dy_2/dx_1)} = \frac{-0{,}286}{0{,}86} = -0{,}33\ (\%\ /\ \text{kg/s})$$ Éq. 6.1

L'équation 6.1 peut être utilisée pour ajuster le gain du contrôleur de consistance automatiquement en implantant une logique qui utiliserait aussi les paramètres de la méthode Lambda (figure 6.9). Cette logique est très utile car les gains en boucle ouverte et les gains relatifs changent selon le point d'opération.

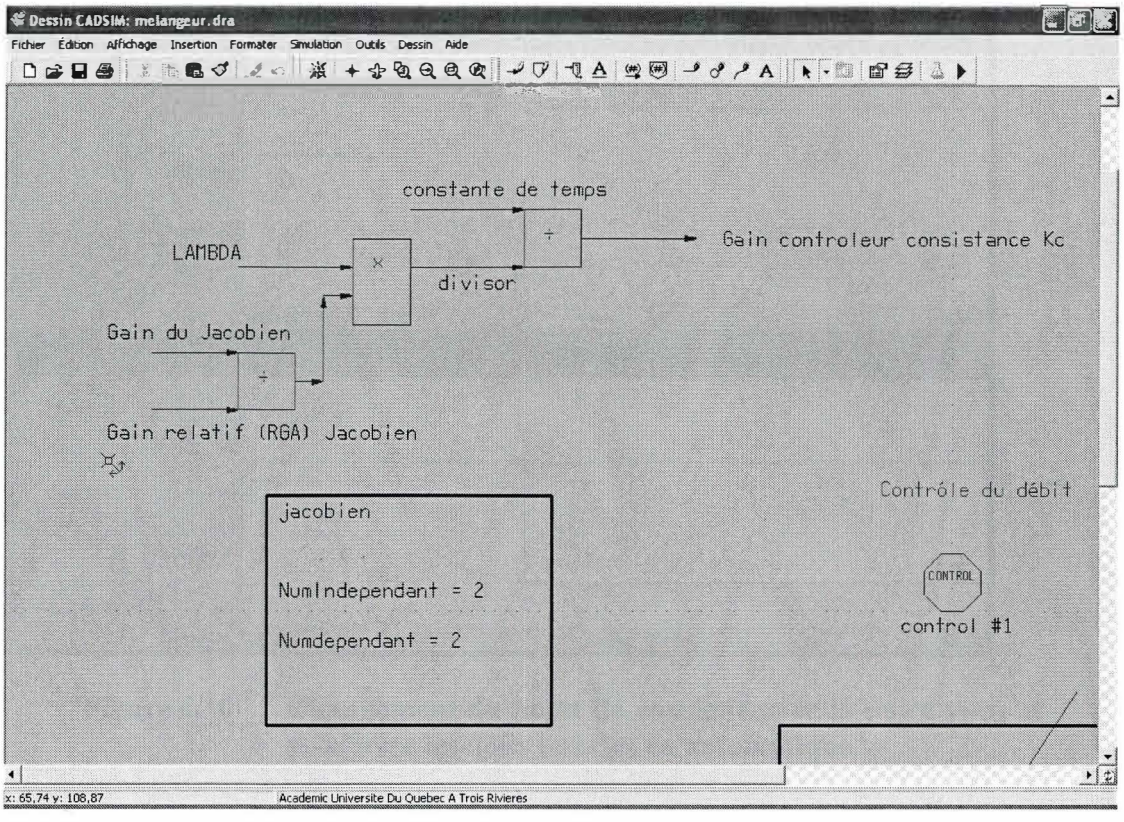


Figure 6.9 Implantation de la logique calculant le gain du contrôleur à partir des résultats du module Jacobien.

6.4.3 Performance du contrôle

La figure 6.10 présente la réponse des contrôleurs lorsqu'on effectue un changement de point de consigne de 60% à 55% avec les deux boucles en mode automatique, on constate que la consistance y_2 a atteint son point de consigne sans un dépassement des variables manipulées x_1 et x_2 comme s'était le cas avec les paires y_1-x_1 et y_2-x_2.

Le choix de la bonne paire variable contrôlée-manipulée s'est avéré déterminant pour améliorer la performance du système du contrôle grâce aux gains relatifs calculés par le module Jacobien.

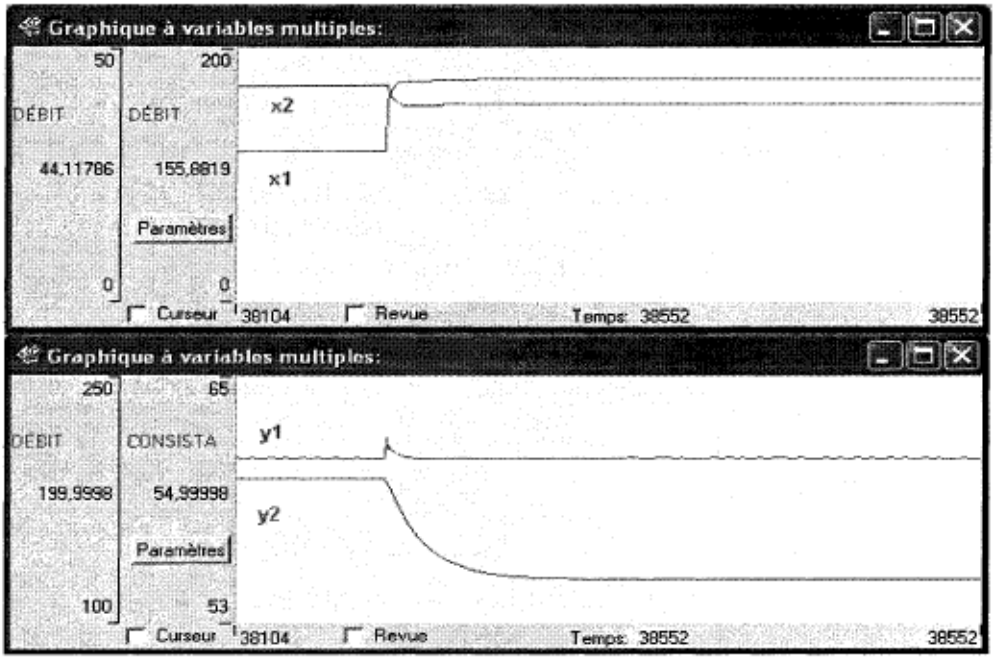


Figure 6.10 **Changement du point de consigne avec la paire y_1-x_2 et y_2-x_1 avec les deux boucles en automatique.**

6.5 Effet du changement du point d'opération

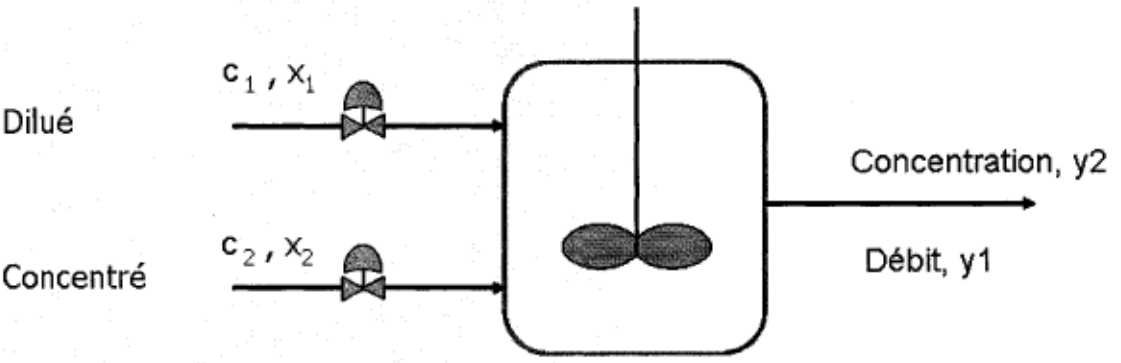


Figure 6.11 **Mélangeur avec manipulation des deux débit x_1 et x_2**

La figure 6.11 montre un mélangeur avec un débit dilué x_1 à une concentration c_1 et un débit concentré x_2 à une concentration c_2. Les deux débits sont mélangés pour atteindre un débit à la sortie y_1 à une concentration y_2.

Les bilans de matières autour du mélangeur nous permettent d'écrire les équations suivantes :

$$y_1 = x_1 + x_2 \qquad \text{Éq. 6.2}$$

$$y_1 * y_2 = x_1 * c_1 + x_2 * c_2 \qquad \text{Éq. 6.3}$$

$$y_2 = \frac{x_1 * c_1 + x_2 * c_2}{x_1 + x_2} \qquad \text{Éq. 6.4}$$

L'équation 6.2 est linéaire, tandis que l'équation 6.4 est non-linéaire.

Le calcul des gains en boucle ouverte équivaut à la dérivée de la variable contrôlée par rapport à la variable manipulée :

$$K_{ij} = \lim_{\Delta x_j \to 0} \frac{\Delta y_i}{\Delta x_j} = \frac{\partial y_i}{\partial x_j} \qquad \text{Éq. 6.5}$$

On peut calculer les gains K_{ij} de la façon suivante :

$$K_{11} = \frac{\partial y_1}{\partial x_1} = 1 \qquad \text{Éq. 6.6}$$

$$K_{12} = \frac{\partial y_1}{\partial x_2} = 1 \qquad \text{Éq. 6.7}$$

$$K_{21} = \frac{\partial y_2}{\partial x_1} = \frac{-x_2(c_2 - c_1)}{(x_1 + x_2)^2} \qquad \text{Éq. 6.8}$$

$$K_{22} = \frac{\partial y_2}{\partial x_2} = \frac{x_1(c_2 - c_1)}{(x_1 + x_2)^2} \qquad \text{Éq. 6.9}$$

On peut constater que les deux gains en boucles ouverte de la variable contrôlée y_2 dépendent des conditions d'opération x_1 et x_2 et donc la MGR sera affectée aussi. Pour

un système 2x2 une simplification du calcul matriciel permet de calculer les gains relatifs à partir des gains en boucles ouverte [7] :

$$\lambda_{ij} = \begin{matrix} y_1 \\ y_2 \end{matrix} \begin{pmatrix} \overset{X1}{\dfrac{K_{11}K_{22}}{K_{11}K_{22}-K_{12}K_{21}}} & \overset{X2}{\dfrac{-K_{12}K_{21}}{K_{11}K_{22}-K_{12}K_{21}}} \\ \dfrac{-K_{12}K_{21}}{K_{11}K_{22}-K_{12}K_{21}} & \dfrac{K_{11}K_{22}}{K_{11}K_{22}-K_{12}K_{21}} \end{pmatrix} \qquad \text{Éq. 6.10}$$

Après simplifications on obtient les gains relatifs suivant :

$$\lambda_{ij} = \begin{matrix} y_1 \\ y_2 \end{matrix} \begin{pmatrix} \overset{X1}{\lambda_{11} = \dfrac{x_1}{x_1+x_2}} & \overset{X2}{\lambda_{12} = \dfrac{x_2}{x_1+x_2}} \\ \lambda_{21} = \dfrac{x_2}{x_1+x_2} & \lambda_{22} = \dfrac{x_1}{x_1+x_2} \end{pmatrix} \text{Éq. 6.11}$$

Les équations théoriques démontrent bien l'effet du changement du point d'opération sur les valeurs des gains et les gains relatifs. Pour le confirmer, on effectue un changement au niveau du point de consigne du contrôleur de consistance de 60% à 10 % et on procède à une perturbation. La figure 6.12 présente les résultats du module Jacobien où les gains relatifs obtenus montrent que les paires souhaitées pour le contrôle à ce nouveau point d'opération sont y_1-x_1 et y_2-x_2.

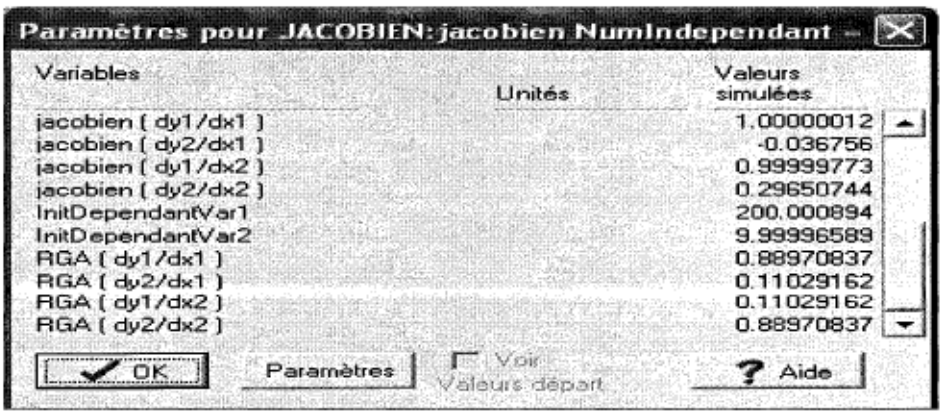


Figure 6.12 Calcul des gains en boucle ouverte et des gains relatifs.

Pour comparer ces résultats au point d'opération à 10 % à ceux obtenus à 60% de consistance, le tableau 6.1 regroupe les gains en boucle ouverte et les gains relatifs obtenus pour ces deux points d'opération.

Tableau 6.1 Effet des conditions d'opération sur la paire de contrôle recommandée

Conditions d'opération	Gains en boucle ouverte	Gains relatifs	Paire de contrôle recommandée
y_2 = 60% , y_1= 200 kg/s	K_{11} = 1 ,K_{12} = 1 K_{21} = -0,286 ,K_{22} = 0,046	λ_{21} = 0,86 λ_{22} = 0,14	y_2-x_1 et y_1-x_2
y_2 = 10%, y_1= 200 kg/s	K_{11} = 1 ,K_{12} = 1 K_{21} = -0,037 ,K_{22} = 0,297	λ_{21} = 0,11 λ_{22} = 0,89	y_2-x_2 et y_1-x_1

Les résultats du tableau 6.1 démontent bien que les gains de procédé et les gains relatifs changent en fonction du point d'opération, l'utilisateur doit changer en conséquence la paire de contrôle recommandée pour minimiser les effets de l'interaction. La logique implantée pour réajuster le gain du contrôleur de la figure 6.9 est indispensable puisque le gain du procédé K_{22} à 10% est 6 fois plus important que celui à 60% de consistance. Donc lorsqu'on change le point d'opération, l'utilisateur demande une perturbation et le module Jacobien réajuste automatiquement le gain du contrôleur.

La figure 6.13 suivante montre un changement du point de consigne du contrôleur de la consistance de 10% à 12% , le gain du contrôleur n'étant pas réajusté automatiquement avec K_c = 91 calculé au point d'opération à 60% de consistance. On constate que la variable manipulée x_2 change excessivement car le gain du contrôleur est grand.

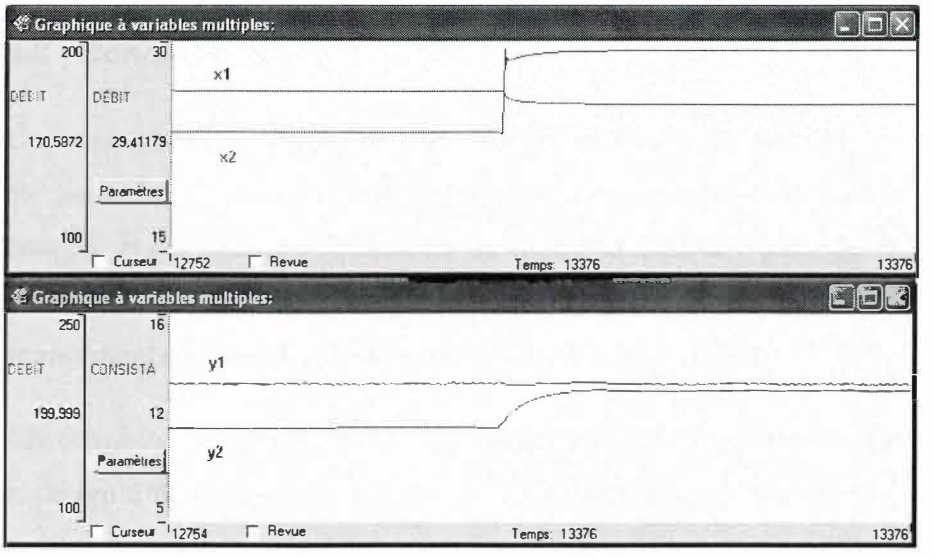


Figure 6.13 Réponse au changement de point de consigne de 10%à 12% sans réajustement du gain contrôleur.

La figure 6.14 montre la réponse à un changement de point de consigne avec ajustement du gain du contrôleur qui vaut 12,64 au lieu de 91. La variable manipulée x_2 change sans dépassement.

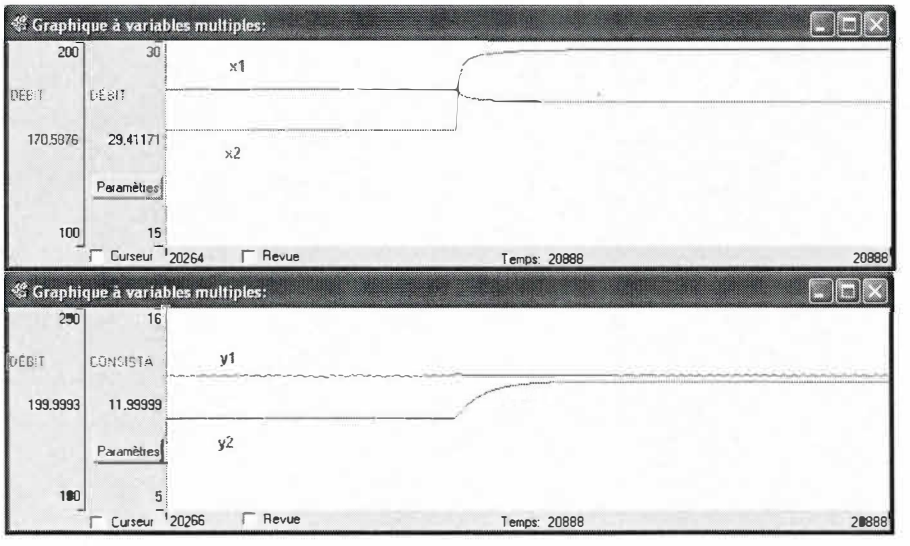


Figure 6.14 Réponse au changement de point de consigne de 10%à 12% avec réajustement du gain contrôleur.

6.6 Conclusions

On a constaté une plus grande performance au niveau du contrôle du mélangeur, ceci a été possible grâce aux calculs du module Jacobien qui évaluent les interactions des boucles, et qui permet à l'utilisateur de choisir la meilleure paire de variables contrôlée-manipulée en plus de réajuster automatiquement le gain du contrôleur lors d'un changement de point d'opération par une simple perturbation.

Les éléments de la matrice MGR du module Jacobien permettent aux ingénieurs de faire le design d'un système de contrôle qui minimise les interactions des boucles. Le module Jacobien est donc un outil de conception efficace lors de l'étape d'ingénierie du système de contrôle.

Chapitre 7 - Conclusions et Recommandations

7.1 Accomplissements

Les trois simulations présentées démontrent l'utilité du module Jacobien pour optimiser la performance du système de contrôle en ajustant le gain du contrôleur.

Pour le cas du concentrateur de liqueur noire, on a constaté que le gain du procédé change beaucoup avec le point d'opération, cette non-linéarité affecte beaucoup la performance et la stabilité du contrôleur. Le module Jacobien a pour objectif de réajuster le gain du contrôleur lorsque l'utilisateur en fait la demande pour un contrôle optimal.

Dans la simulation du contrôle de lavage de pâte chimique, la teneur en solides dissous au dernier stade de lavage affecte la qualité (propreté de la pâte) et les coûts d'opération, une teneur de 16,4 kg/tonne pâte est optimale pour minimiser les coûts d'opération. Le contrôle de l'eau utilisée pour le lavage est déterminant pour maintenir la teneur des solides dissous le plus près possible du point de consigne optimal. Trois types de contrôle ont été évalués, la programmation des gains qui utilise les gains optimaux du Jacobien offre un meilleur contrôle des solides dissous par rapport au contrôle ordinaire et manuel avec une meilleur réduction de la variabilité et des coûts d'opérations minimisés.

Pour le contrôle multivariable, le calcul de la matrice des gains relatif a permis une meilleure performance du contrôle du mélangeur avec le choix des paires de variables contrôlées-manipulées qui minimise les interactions des boucles. Le module Jacobien facilite la tâche à l'utilisateur puisqu'il peut calculer la MGR à différents points d'opération avec une grande rapidité, il serait donc d'une grande utilité dans un système de contrôle où plusieurs variables manipulées et contrôlées sont en jeu.

Le module Jacobien qui a été développé et programmé dans le cadre de ce projet de maîtrise offre aux utilisateurs de Cadsim Plus un outil efficace pour le calcul des gains et la MGR. Le calcul des gains ou les dérivées partielles pourrait être utilisé comme outils d'optimisation mathématique.

7.2 Améliorations possibles et travaux futurs

Le module Jacobien développé offre aux utilisateurs la possibilité d'introduire le temps de stabilisation et le facteur de perturbation qu'ils souhaitent en fonction du type du procédé de la simulation. Le point fort de cette approche est sa grande flexibilité au niveau du choix des paramètres du module Jacobien. Par contre, le point faible c'est que le temps de stabilisation et le facteur de perturbation sont constants pour toutes les variables à chaque simulation, l'utilisateur doit donc choisir de façon appropriée un temps de stabilisation assez grand pour satisfaire toutes les variables ainsi qu'un facteur de perturbation convenable à chaque variable manipulée.

Une amélioration des paramètres du module est possible grâce à l'automatisation du calcul du «bump factor» et du «settling time» où ces paramètres sont évalués pour chaque paire de variables manipulées et contrôlées, ce qui permet un calcul plus précis du gain et ainsi la matrice des gains relatif et réduire de façon optimale le temps de stabilisation pour chaque variable dépendante y_i.

Bibliographie

1 Sell, Nancy J. (1995) "Process control fundamentals for the pulp & paper industry" *TAPPI PRESS,* ISBN 0-89852-294-3.

2 P.Hynninen (1998) "Papermaking science and technology : Environmental control", *Technical Association of the Pulp and Paper Industry,* ISBN 952-5216-19-5

3 Stuart Bennett (1996) "A brief history of automatic control" *IEEE Control Systems Magazine* 16/3:17-25.

4 A.Rossiter I.S.Choi, J.Fleming (2007) "Looper and tension control in hot rolling mills: A survey" *Journal of Process Control* 17/6: 509-521.

5 J.Castro, F.J.Doyle (2004) "A pulp mill benchmark problem for control: application of plant wide control design" *Journal of Process Control* 14/3: 329-347.

6 Dale E. Seborg, Thomas F. Edgar, Duncan A.Mellichamp (2004) "*Process dynamics and control*", 2nd Edition, John Wiley & Sons, ISBN 0-471-00077-9.

7 Carlos A. Smith, Armando B. Corripio (1997) "*Principles and practice of automatic process control*", 2nd Edition, J.Wiley, ISBN 0-471-57588-7.

8 W. L. Bialkowski (1998) "Mill audits can cut costs by reducing control loop variability" *Pulp & Paper* 72/12: 79-84 .

9 T.Hägglund, A.Ingimundarson, S.Schwaber (2005) "Closed-loop performance monitoring using loop tuning" *Journal of Process Control* 15/2: 127-133.

10 R.Anandanatarajan, M.Chidambaram, T.Jayasingh (2006) "Limitation of PI controller for first order nonlinear process with dead time" *ISA Transactions*, 45/2: 185-199.

11 Thomas E. Marlin (2000) "*Process control: designing processes and control systems for dynamic performance* ", 2nd Edition, McGraw-Hill, ISBN 0-07-039362-1.

12 J.Gao, H.Budman (2005) "Design of robust gain scheduled PI controllers for nonlinear processes" *Journal of Process Control* 15/7: 807-817.

13 J.Doyle, S.Kwatra, S.Schwaber (1998) "Dynamic gain scheduled process control" *Chemical Engineering Science* 53/15: 2675-2690.

14 K. Åstrom , T. Hagglund (1995) "*PID controllers : theory, design, and tuning*", 2nd Edition, International Society for Measurement and Control, ISBN 1-55617-516-7.

15 D.Lillfors, C. Basset, R. Akkerman, and S. Kovac (2003) *"Blandin reduces variability in pulp brightness with adaptative controller" Pulp & Paper* 77/1: 48-50.

16 F.Garelli, R.J.Mantz, H.De Battista (2006) "Limiting interaction in decentralized control of MIMO systems" *Journal of Process Control* 16/5: 473-483.

17 L.Al-Awami, Y.Sidrak (1998) "Understanding the dynamic behaviour of Kamyr digesters" *ISA Transactions*, 37/1: 53-64.

18 P.Samuelsson, B.Halvarsson, B.Carlsson (2005) "Interaction analysis and control structure selection in wastewater treatment plant model " *IEEE Transactions on Control Systems Technology* 13/6: 955-964.

19 Department of Chemical Engineering, University of Edinburgh, "*Module 6: Multivariable Systems" http://eweb.chemeng.ed.ac.uk/courses/control/restricted /course/advanced/outline6.html.*

20 T.F.Edgar, J.Lee (2002) "Interaction measure for decentralized control of multivariable processes" *American control conference* vol: 1 : 454-458 .

21 M.J. Willis (1999) "*Multivariable control : An Introduction*" University of Newcastle upon Tyne, http://lorien.ncl.ac.uk/ming/mloop/MULTIVAR.pdf

22 Armando B. Corripio (2001) "*Tuning of industrial control systems*", 2nd Edition, Instrument Society of America, ISBN 1-55617-713-5.

23 Bequette B. Wayne (2003) "*Process control : modeling, design, and simulation*", Prentice Hall, ISBN 0-13-353640-8.

24 I.Lama, M.Perrier, P.Stuart (2006) "*Controllability analysis of a TMP-newsprint refining process*" Pulp & Paper Canada 107/10: 212-216.

25 L.Laperrière (2005) "*Introduction au logiciel de simulation dynamique CADSIM PLUS* ".

26 Aurel Systems (2006) "*CADSIM PLUS DLM Programmers Manuel* ".

27 L.Wasik, D.Nelson, G.Mittet (2000) "Controlling Brown stock washing during production rate changes" *TAPPI Journal* 83/3.

28 L.Wasik, G.Desaulniers, G.Mittet (1996) "*Controlling Brown stock washers for optimum performance* ". Pulp Washing , Canadian Pulp & Paper Association, TAPPI & CPPA ,pages : 95-101.

29 G.Smook (1998) "*Manuel du technicien et de la technicienne en pâtes et papiers*", 2ème édition, Centre collégial de développement de matériel didactique, ISBN 2-89470-063-6.

30 J.Gullichsen, C.Fogelholm "*Papermaking science and technology: chemical pulping A*", TAPPI Press (2000), ISBN 952-5216-06-3.

31 P.Hart (2005) " Brown Stock Washing Short Course : Washing" *TAPPI Press* .

32 US Environmental Protection Agency " *Regulatory Bases for Effluent Limitations Guidelines and Standards for Subparts B and E?"* www.epa.gov/waterscience/pulppaper/permitguide/chapter 5.pdf

33 B.Gough, J.T.Kay (1996) "Kraft continuous digester effective alkali control" *Pulp & Paper Industry Technical Conference*, pages: 32-38, ISBN 0-7803-3148-6.

34 W.Iverson "*Washing out energy costs" http://www.automationworld.com/.*

Annexe 1

Propriétés du contrôleur PID

a) Contrôle proportionnel

L'objectif du contrôle en rétroaction est de réduire l'erreur entre la consigne $y_{sp}(t)$ et la mesure $y_m(t)$ de la variable à contrôler. La grandeur du signal d'erreur e(t) est exprimé par l'équation suivante :

$$e(t)=y_{sp}(t)-y_m(t)$$

En mode proportionnel, la sortie du contrôleur est proportionnelle à l'erreur e(t).

$$p(t) = \overline{p} + K_c\, e(t)$$

Avec :

p(t) : Sortie du contrôleur

$\overline{p}$: Valeur du bias (en régime permanent lorsque e = 0)

K_c : Gain du contrôleur

Le gain du contrôleur peut être ajusté de façon à augmenter ou diminuer la sensibilité de la sortie du contrôleur face à une déviation entre la consigne et la valeur mesurée de la variable à contrôler. La valeur du bias est déterminée lorsque la variable manipulée a atteint sa valeur nominale en régime permanent quand l'erreur est nulle.

La fonction de transfert exprimée en variable de déviation du contrôleur proportionnel est exprimée par l'équation suivante :

$$\frac{p'(s)}{e(s)} = K_c$$

Le contrôle proportionnel est simple puisqu'il y a un seul paramètre à ajuster (K_c). Cependant il ne permet qu'une diminution de l'erreur en régime permanent sans l'éliminer lors d'un changement de point de consigne ou après une perturbation soutenue, c'est ce qu'on appelle un « offset ». Pour expliquer cette erreur due à l'offset, considérons le cas d'un réservoir où l'objectif de contrôle est de maintenir le niveau du liquide à un point de consigne donné en manipulant une vanne à la sortie du réservoir. Si on considère qu'avec une ouverture de vanne de 50%, le niveau est maintenu au point de consigne, l'équation du contrôleur devient : $p(t) = 50\% + K_c\, e(t)$

Maintenant si le débit d'entrée au réservoir est augmenté, le niveau au réservoir va augmenter aussi, alors le contrôleur doit ouvrir plus sa vanne à la sortie (disons à 60%) pour ramener le niveau au point de consigne. Avec l'équation du contrôleur proportionnel, l'erreur ne pourrait jamais atteindre zéro pour que la sortie soit égale à 60%. Si K_c vaut 1, l'erreur est de 10%, on l'appelle l'offset. Avec un K_c plus élevée, on pourrait diminuet l'offset, mais il y a des risques d'instabilité du contrôleur [7]. Pour régler ce problème il faut changer la valeur du bias, ou tout simplement ajouter un contrôle intégral.

Dans certains cas de contrôle de niveau où «l'offset» est toléré, le controle proportionnel est suffisant, car l'objectif est de maintenir un niveau acceptable pour éviter le débordement ou de vider complètement le réservoir.

b) Contrôle intégral

En mode de contrôle intégral, la sortie du contrôleur dépend de l'intégrale du signal d'erreur en fonction du temps :

$$p(t) = \bar{p} + \frac{1}{\tau_I}\int_0 e(t)\,dt$$

τ_I est un paramètre ajustable du contrôleur PID, il représente la constante de temps. L'action intégrale est utilisée pour éliminer «l'offset» ce qui permet à la variable contrôlée d'atteindre le point de consigne. L'action intégrale n'est significative que

lorsque l'erreur est persistante dans le temps, tandis que l'action proportionnelle est immédiate. Pour cette raison, le mode de contrôle intégrale et utilisé conjointement avec le mode proportionnel. La fonction de transfert pour un contrôleur PI est la suivante :

$$\frac{P'(s)}{E(s)} = K_c\left(1+\frac{1}{\tau_I s}\right)$$

Pour un échelon unitaire de l'erreur e(t), on a un changement instantané de la sortie du contrôleur dû à l'action proportionnelle. Lorsque t vaut τ_I l'action intégrale contribue de la même quantité à la sortie du contrôleur que l'action proportionnelle.

Le désavantage du mode intégral réside dans la tendance oscillatoire de la variable contrôlée, la réduction de la stabilité de la boucle de contrôle et le phénomène du «reset windup». Néanmoins le contrôle PI est très majoritairement utilisé en industrie car il permet de réduire l'offset [6].

Le phénomène du «reset windup» est souvent rencontré en contrôle des procédés pour des contrôleurs utilisant l'action intégrale lors d'un grand changement au niveau du point de consigne ou de la perturbation. Dans ce cas de figure, le terme intégral devient très élevé de telle façon que la sortie du contrôleur dépasse la limite physique permise de la variable manipulée (0-100% d'ouverture pour une vanne) et peut se saturer à -15% ou 115%. Le cas échéant, même si la variable contrôlée a atteint la consigne et que la sortie du contrôleur commence à diminuer, la variable manipulée restera à sa valeur de saturation jusqu'à ce que la sortie du contrôleur revienne en dessous de la valeur de saturation, ce qui produit un dépassement. Pour faire face à ce problème, les contrôleurs industriels sont dotés «d'antireset windup» pour suspendre l'intégration lorsque le contrôleur est en saturation. [7].

c) Contrôle dérivatif

L'action dérivative permet de prévoir le comportement futur du signal d'erreur en analysant son taux de changement. L'équation suivante détermine l'action dérivative :

$$p(t) = \bar{p} + \tau_D \frac{de(t)}{dt}$$

Le troisième paramètre d'ajustement d'un contrôleur PID est le temps dérivatif τ_D. Le mode dérivatif est associé au contrôle proportionnel et intégral dans le but de contrer la tendance oscillatoire du mode intégral en plus de réduire le temps requis pour atteindre la valeur de la consigne. Cependant, pour un système contenant du bruit (grandes fluctuations) dans les mesures, l'action dérivative devient large, et par conséquent elle amplifie le bruit. Pour remédier à ce problème, il faut filtrer les mesures.

Ajustement du contrôleur PID

a) Méthode de Ziegler-Nichols (méthode d'oscillation continue)

Cette méthode était publiée par Ziegler et Nichols en 1942, elle est utilisée pour une boucle fermée (contrôleur en mode automatique). Le but est d'avoir une oscillation constante et prolongée de la variable contrôlée. La procédure utilisée est la suivante [6] :

Étape 1 : Une fois que le procédé a atteint le régime permanent, on élimine l'action intégrale du paramètre τ_I (la plus grande valeur possible) et l'action dérivative du paramètre τ_D (valeur nulle).

Étape 2 : Configurer le gain du contrôleur K_c à 0,5 et placer le contrôleur en mode automatique.

Étape 3 : Faire un petit changement instantané du point de consigne (setpoint) pour que la variable contrôlée change sa valeur puis augmenter la valeur du gain K_c progressivement jusqu'à ce qu'on atteigne une oscillation stable et continue. La valeur de K_c qui produit cette oscillation représente le **gain ultime K_{cu},** quant à la période **P_u** est appelée **période ultime.**

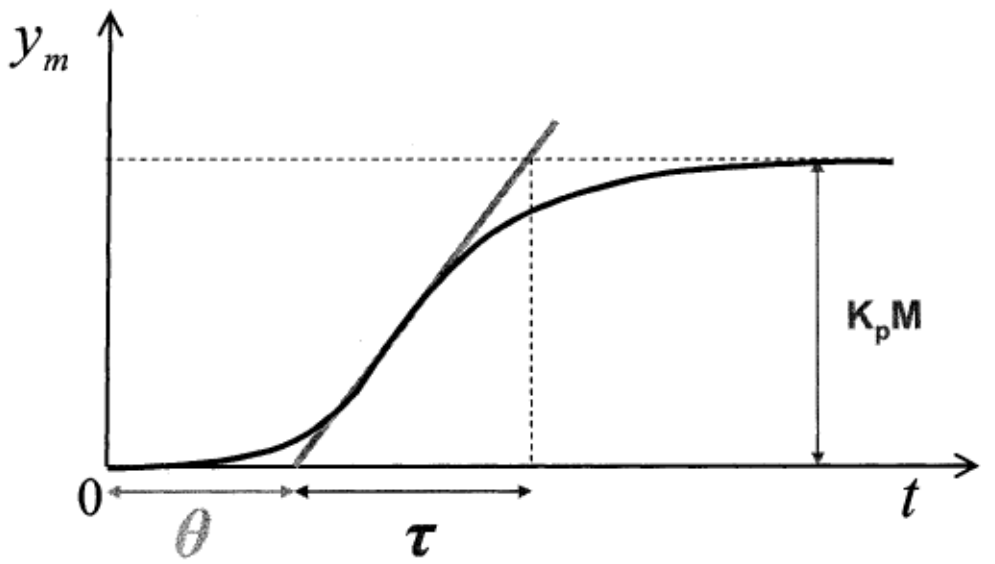
Étape 4 : Calcul des paramètres du contrôleur en utilisant les formules de Ziegler et Nichols ou encore celle Tyreus-Luyben résumés dans le tableau si dessous.

Ajustement des paramètres du contrôleur par la méthode d'oscillation continue :

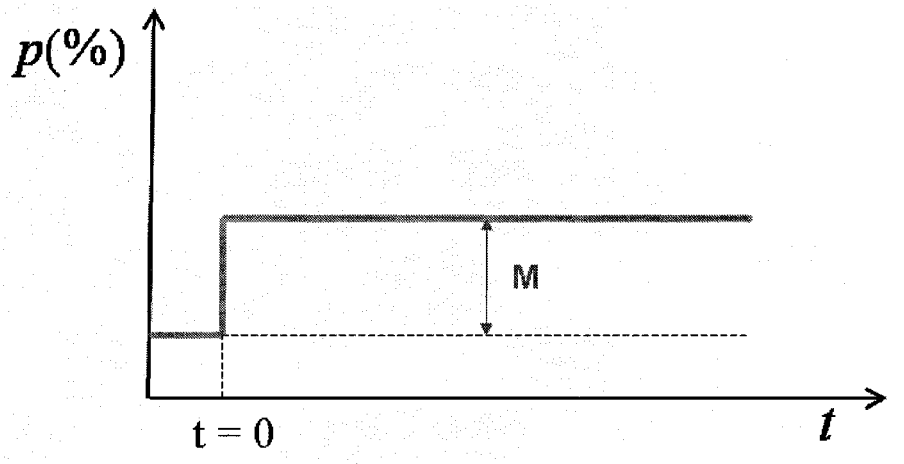
Ziegler-Nichols	K_C	τ_I	τ_D
P	0,5 K_{cu}	-----	-----
PI	0,45 K_{cu}	P_u /1.2	-----
PID	0,6 K_{cu}	P_u /2	P_u/8
Tyreus-Luyben	**K_C**	**τ_I**	**τ_D**
PI	0,31 K_{cu}	2,2	-----
PID	0,45 K_{cu}	2,2	/6,3

b) Méthode de Ziegler-Nichols en boucle ouverte

Ziegler et Nichols ont aussi développés une méthode d'ajustement en boucle ouverte basée sur un test échelon «Step test». Le contrôleur est mis en mode manuel et on procède à un petit changement échelon dans la sortie du contrôleur (3 à 5%) et on enregistre la réponse du procédé. K_P représente le gain du procédé, τ est la constante de temps et θ le délai qui est souvent relié au transport.



Paramètres d'un système de 1er ordre avec délai.



Gain du procédé

C'est un paramètre important du procédé, il mesure la sensibilité de la variable de sortie du procédé par rapport à celle de l'entrée :

$$K_P = \frac{changement\ de\ la\ sortie}{changement\ de\ l'\ entrée} = \frac{\Delta y}{\Delta M}$$

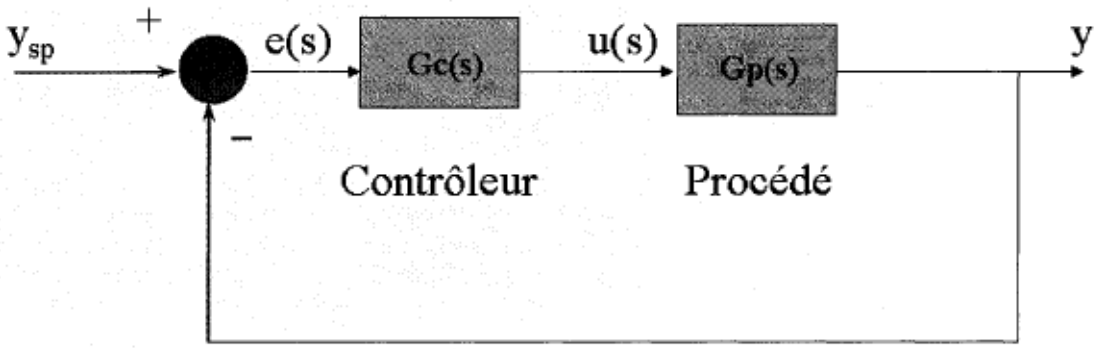
Le changement au niveau de la variable contrôlée «y» est calculé après que cette dernière atteigne le régime permanent. Le gain du procédé dépend des propriétés physiques et du point d'opération, ses unités sont exprimés par le pourcentage de la sortie du transmetteur par rapport à celle du contrôleur [7].

Paramètres du contrôleur ajustés par la méthode Ziegler et Nichols en boucle ouverte[1] :

Ziegler-Nichols	K_C	τ_I	τ_D
P	$\tau / \theta K_p$	---	---
PI	$0.9\ \tau / \theta K_p$	$3.3\ \theta$	---
PID	$1.2\ \tau / \theta K_p$	$2.0\ \theta$	$0.5\ \theta$

c) Méthode Lambda

Cette méthode est aussi basée sur le modèle du procédé. Pour des modèles simple (1er ordre , 2ème ordre avec ou sans délai) les paramètres du gain, la constante de temps et le délai, sont obtenus par des test échelons «perturbation test» en boucle ouverte [1].



On veut trouver les paramètres du contrôleur de façon à ce que :

$$G_c(s)=\frac{1}{G_p(s)}\bullet\frac{1}{\lambda s}$$

Le paramètre « λ » correspond à la constante de temps désirée en boucle fermée.

La fonction de transfert de la boucle ouverte :

$$G_{OL}(s)=G_c(s)\bullet G_P(s)=\frac{1}{\lambda s}$$

En boucle fermée, on peut constater que la relation entre la sortie « y » et la consigne est une fonction de premier ordre avec un gain de 1 :

$$\frac{Y}{Y_{sp}}=\frac{G_{OL}(s)}{1+G_{OL}(s)}=\frac{1}{1+\lambda s}$$

La fonction de transfert correspondant à un PID :

$$G_c(s) = K_C\left[1+\frac{1}{\tau_I s}+\tau_D s\right]$$

Pour un procédé représentant un comportement de premier ordre on a :

$$G_P = \frac{K_p}{\varpi + 1}$$

$$G_c(s)=\frac{\tau s+1}{K_P}\bullet\frac{1}{\lambda s}$$

Dans la méthode d'ajustement Lambda, les paramètres du PID sont sélectionnés de façon à ce que les zéros de la fonction de transfert du contrôleur neutralisent les pôles de la fonction de transfert du procédé G_p. Dans le cas de boucles de débit de premier ordre, l'action intégrale devra être négligée.

$$G_c(s) = K_C\left[1+\frac{1}{\tau_I s}\right] = K_C\bullet\frac{\tau_I s+1}{\tau_I s}$$

On peut estimer les paramètres du contrôleur :

$$K_C = \frac{\tau}{K_p \lambda}$$

$\tau_I = \tau$

Le gain K_p est souvent en unité d'ingénierie, il faut le convertir en pourcentage en se basant sur la largeur de la bande de mesure de l'instrument «Span» :

$$K'_P = \frac{\frac{\Delta y}{span(y)}}{\frac{\Delta(u)}{span(u)}} = \frac{\Delta y}{\Delta u}\bullet\frac{1}{SR} = K_P\bullet\frac{1}{SR}$$

où :

SR : ratio de normalisation ($SR = \frac{Span(y)}{Span(u)}$)

En substituant on trouve:

$$K_C = \frac{\tau}{K'_p \lambda} = \frac{\tau}{K_p \lambda} \cdot SR$$

$$\frac{K_C}{SR} = \frac{\tau}{K_p \lambda}$$

Paramètres du contrôleur ajustés par la méthode Lambda [1] :

Modèle du procédé	**K_C / SR**	**τ_I**	**τ_D**
$\frac{K_p}{\tau s + 1}$	$\frac{\tau}{K_p \lambda}$	τ	-----
$\frac{K_p e^{-\theta s}}{\tau s + 1}$	$\frac{\tau}{K_p (\lambda + \theta)}$	τ	-----
$\frac{K_p e^{-\theta s}}{\tau s + 1}$	$\frac{\tau}{K_p (\lambda + \frac{\theta}{2})}$	τ	θ /2
$\frac{K_p e^{-\theta s}}{(\tau_1 s + 1)(\tau_2 s + 1)}$	$\frac{\tau_1}{K_p (\lambda + \theta)}$	τ_1	τ_2
$\frac{K_p}{s}$	$\frac{2}{K_p \lambda} = \frac{100}{ALV}$	2λ	-----
$\frac{K_p e^{-\theta s}}{s}$	$\frac{2\lambda + \theta}{K_p (\lambda + \theta)^2}$	$2\lambda + \theta$	-----

Le tableau ci-dessus résume les paramètres d'ajustement du contrôleur avec la méthode Lambda, en se basant sur le modèle de chaque procédé.

L'utilisateur doit spécifier la valeur de « λ » qui détermine la constante de temps de la boucle fermée. Une valeur de « λ » équivalente à 2 fois la valeur de la constante de temps du procédé « τ » est souhaitable pour les procédés non-intégrateurs. Une grande valeur de « λ » donne une réponse lente et une valeur plus faible augmente l'instabilité de la boucle de contrôle et génère une action excessive du contrôleur [1].

Cette méthode d'ajustement est très utilisée dans l'industrie papetière, elle permet une réponse non-oscillatoire lors d'un changement de point de consigne ou après une perturbation [8].

Annexe 2

Listing du programme Jacobien développé dans cette recherche

```
/* Copyright @ 1995 Aurel Systems, Inc. All rights reserved.                    */

/* File:                JACOBIEN.C      (Copy and rename to your own FILENAME.C)
        */
/* Author(s):      MOHAMED CHBEL
                                                                                */
/* Created:             Date
                                                                                */
/* Modified:
                                                                                */
/*                                              2007/02/13:     added non-flow component capability
                        */

/* Notes:          This sample DLM source file is based on the CADSIM Plus      */
/*                              DIVIDER process module which was written by T. Casavant     */
/*                              on 95/02/05.  You can use this DLM module as a basis for    */
/*                              your own original DLM modules.  Copy this SAMPLE.C file to  */
/*                              a new name, and modify it to your own specifications.
        */
/*                              Use the "Building Your Own Process Unit DLM" documentation  */
/*                                      as a guide.  You must then add your new DLM source file to*/
/*                                      the Borland project file SAMPLIB.IDE located in your
        */
/*                                      CSPLUS\MAKE directory.  Use the SAMPLE.C entry as an
        */
/*                                      example and see your Borland C++ documentation for more
        */
/*                                      information.
                                                                                */

// Following 2 precompiled header lines must appear at the top of all files.
#include "dlm_pch.h"
#pragma hdrstop
#include "math.h"
#include "cntrl.h"
#include "dsply.h"
#include "strm.h"
#include "unit.h"
#include "physname.h"
#include "mod.h"


                /*=============================================================*/
                /*
                                                                                */
                /*                      Description, Requirements and Recognition
                */
                /*
                                                                                */
                /*=============================================================*/

#define kModuleDescription   "CALCUL JACOBIEN"
/* !!!!!!!!!!!!!!!!!! DO NOT CHANGE THE FOLLOWING LINE !!!!!!!!!!!!!!!!!! */
#define kVendorNumber                kVendorAurelCadsimPlus
/* !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! */
#define kFileTag                                JACOBIEN

static Qchar        kKeywordNames[] =
 {
        "JACOBIEN",
    "JACOBIAN",
```

```
        END_OF_LIST
};

static Qchar        kStreamRequirements[] =
{
    END_OF_LIST
};

static T_NameValue          kFreeUnitVariableTable[] =
{
  { "BUMP",       0 },
  { "SETTLING TIME",  10 },
  { "BUMP FACTOR",  0.1 },

        /* The following are varying: */
        END_OF_NAMES_VALUES
};

enum EFreeVarIndex  //used in perform()
{
        indexBUMP,          /* 0 */
        indexSETTLING_TIME,         /* 1 */
    indexBUMP_FACTOR    /* 2 */
}; /* EFreeVarIndex */

static Qchar        kCalcVariables[] =
{

        END_OF_LIST
};



static Qchar        kQualifiers[] =
{
        DYNAMIC_NAME,
        END_OF_LIST
};

enum EFreeUnitVarIndex  //used in GetUnitFreeInfo
{
        kBUMP,  /* 0 */
        kSETTLING_TIME, /* 1 */
    kBUMP_FACTOR,       /* 2 */
    kVarIndependante,
    kVarDependante,
    kNoMore
};

static EFreeUnitVarIndex    gWhichSpec;
static TIndex                       gIndex;
TIndex i_Jacobien , j_Jacobien,i_rga , j_rga , VarDepIndex;

Begin_private_data(kFileTag)
TIndex    num_independant , num_dependant ,Taille_Jacb;
int iteration ,independant_index , total_count , iteration_bump;
int calculJacobien_index;

//pour accéder les variables globales à ce fichier avec local->
//double surface_base;
//double hauteur;

End_private_data(kFileTag)

Standard_unit_description
(kFileTag,                                                  /* source file name */
 kVendorNumber,                                 /* vendor number */
 kModuleDescription,                            /* module description */
 kKeywordNames,                                 /* module keywords */
 kFreeUnitVariableTable,    /* names of unit free variables/default values */
```

```
kCalcVariables,                         /* names of unit calc variables */
kStreamRequirements,                    /* names of required variables */
kQualifiers,                                    /* names of unit qualifiers */
kGENERIC,                                               /* what kind of unit */
0,                                                              /* minimum number of inputs */
0,                                                              /* maximum number of inputs */
0,                                                              /* minimum number of outputs */
0,                                      /* maximum number of outputs */
false,                                  /* outlet variables are at a point */
false,                                  /* supports multiple stream definitions */
false,                                  /* input streams have special names */
false,                                  /* output streams have special names */
false,                                  /* split ratios allowed */
false,                                  /* auxiliary variables are normal */
true,                                   /* number of free unit variables varies */
false,                                  /* can pass networked flows */
kPopup_standard)                /* kind of unit pop-up */

                /*============================================================*/
                /*                                                            */
                /*                      Creation                              */
                /*                                                            */
                /*============================================================*/

/*------------------------------ SetUp ---------------------------------*/
Standard_SetUp_function

                /*============================================================*/
                /*                                                            */
                /*                      Unit Matching                         */
                /*                                                            */
                /*============================================================*/

/*------------------------------ GetInputDefaultName -------------------*/
Standard_GetInputDefaultName_function

/*------------------------------ GetOutputDefaultName -------------------*/
Standard_GetOutputDefaultName_function

        /* GetOutputDefaultName */

                /*============================================================*/
                /*                                                            */
                /*                      Initialization                        */
                /*                                                            */
                /*============================================================*/

/*------------------------------ PrepareForSpecifications ---------------*/
Standard_PrepareForSpecifications_function

/*------------------------------ PrepareGetUnitFreeInfo -----------------*/
//Standard_PrepareGetUnitFreeInfo_function
 PrepareGetUnitFreeInfo_function
 {
        Pchar valueStringA ,valueStringB;
    TIndex      count;
        RecognizeAllArguments(PrepareGetUnitFreeInfo)
        if (verbose)
                PrintSelf("- preparing to get next unit free info for ", " -\n");

        valueStringA = GetCopyOfKeywordValueFromUnitName("NumIndependant");
```

```
valueStringB = GetCopyOfKeywordValueFromUnitName("NumDependant");
        if (valueStringA)
         {
                local->num_independant = atoi(valueStringA);
                if (local->num_independant < 0)
                        local->num_independant = 0;
                FreeMemory(valueStringA);
         }
         else
                local->num_independant = 0;


 if (valueStringB)
        {
                local->num_dependant = atoi(valueStringB);
                if (local->num_dependant < 0)
                        local->num_dependant = 0;
                FreeMemory(valueStringB);
        }
        else
                local->num_dependant = 0;

count = local->num_independant +local->num_dependant + 3 ;
gWhichSpec = kBUMP;
        gIndex = 0;

        return count ;



} /* PrepareGetUnitFreeInfo */

 /*------------------------------ GetNextUnitFreeInfo ----------------------*/
//Standard_GetNextUnitFreeInfo_function
 //Cette fonction renvoi le texte a afficher en mode specification
GetNextUnitFreeInfo_function
 {
                Qchar           result = NULL;
        TIndex  index;

        RecognizeAllArguments(GetNextUnitFreeInfo)
        if (verbose)
                PrintSelf("- getting next unit free info for ", NULL);

    switch (gWhichSpec)
        {
                case kBUMP:
                        result = "BUMP";
                        gWhichSpec = kSETTLING_TIME;
                        break;

                case kSETTLING_TIME:
                        result = "SETTLING TIME";
                        gWhichSpec =  kBUMP_FACTOR;
                        break;

        case kBUMP_FACTOR:
                        result = "BUMP FACTOR";
                        gWhichSpec =  kVarIndependante;
                        break;

        case kVarIndependante:

         index = gIndex++ ;
                        strcpy(gTempBuff, "VarIndependante");
                        strcat(gTempBuff, StringFromIndex(NULL, 0, index + 1));
                        if (strlen(gTempBuff) > maxStringLen)
                                *(gTempBuff + maxStringLen) = E_O_S;
```

```
                        result = gTempBuff;
                        if (gIndex == local->num_independant)
                {
                                gWhichSpec = kVarDependante;
                  gIndex =0;
                 }
                        break;


        case kVarDependante:

        index = gIndex++ ;
                        strcpy(gTempBuff, "VarDependante");
                        strcat(gTempBuff, StringFromIndex(NULL, 0, index + 1));
                        if (strlen(gTempBuff) > maxStringLen)
                                *(gTempBuff + maxStringLen) = E_O_S;
                        result = gTempBuff;
                        if (gIndex == local->num_dependant)
                {
                                gWhichSpec = kNoMore;
                                gIndex = 0;
                 }
                        break;


      } //end switch


        if (verbose)
         {
                if (result)
                        PrintStrings(" = ", result, " -\n");
                else
                        PrintString(" = NULL -\n");
     }
        return result;
 } /* GetNextUnitFreeInfo */

 /*------------------------------- PrepareGetDeferredVarInfo ------------*/
PrepareGetDeferredVarInfo_function
  {


  RecognizeAllArguments(PrepareGetDeferredVarInfo)
        if (verbose)
                PrintSelf("- preparing to get next deferred var info for ", " -\n");
        gIndex = 0;
    local->Taille_Jacb = 0;
    i_Jacobien = 1;
    j_Jacobien = 1;
    i_rga = 1;
    j_rga = 1;
    VarDepIndex = 0;
        return (local->num_independant * local->num_dependant)*2 + local->num_independant + local-
>num_dependant;
   }
/*------------------------------- GetNextDeferredVarInfo ----------------*/
GetNextDeferredVarInfo_function
{
     Qchar result;


    local->Taille_Jacb = (local->num_independant * local->num_dependant)+ local->num_independant ;
    gIndex++ ;
        RecognizeAllArguments(GetNextDeferredVarInfo)
        if (verbose)
                PrintSelf("- getting next deferred var info for ", NULL);
```

```
// Affichage des valeurs initiales des variables indépendantes avant le bump
  if (VarDepIndex < local->num_dependant &&  gIndex > local->Taille_Jacb)

          {
                strcpy(gTempBuff, "InitDependantVar");
                strcat(gTempBuff, StringFromIndex(NULL, 0, ++VarDepIndex));
                if (strlen(gTempBuff) > maxStringLen)
                        *(gTempBuff + maxStringLen) = E_O_S;
                result = gTempBuff;

     }

 // Affichage des Current Independant Variables qui sont la valeur Bumpé de la variable indépendante

      if (gIndex < local->num_independant+1)
       {
                strcpy(gTempBuff, "CurrentIndependantVar");
                strcat(gTempBuff, StringFromIndex(NULL, 0, gIndex));
                if (strlen(gTempBuff) > maxStringLen)
                        *(gTempBuff + maxStringLen) = E_O_S;
                result = gTempBuff;
   }

// Affichage de la matrice du Jacobien (i,j)


else if (gIndex < local->Taille_Jacb+1)
       {

      if ( i_Jacobien < local->num_independant+1 )
      {
         if (j_Jacobien < local->num_dependant+1 )
         {
                    strcpy(gTempBuff, "Jacobien ( dy");
          strcat(gTempBuff, StringFromIndex(NULL, 0,
                    j_Jacobien++ ));
          strcat(gTempBuff,  "/dx");
                strcat(gTempBuff, StringFromIndex(NULL, 0,
                    i_Jacobien ));
          strcat(gTempBuff,  " )");
                if (strlen(gTempBuff) > maxStringLen)
                        *(gTempBuff + maxStringLen) = E_O_S;
                    result = gTempBuff;
         }
         else
         {
           j_Jacobien =1  ; i_Jacobien ++ ;

          strcpy(gTempBuff, "Jacobien ( dy");
          strcat(gTempBuff, StringFromIndex(NULL, 0,
                    j_Jacobien++ ));
          strcat(gTempBuff,  "/dx");
                strcat(gTempBuff, StringFromIndex(NULL, 0,
                    i_Jacobien ));
          strcat(gTempBuff,  " )");

                if (strlen(gTempBuff) > maxStringLen)
                        *(gTempBuff + maxStringLen) = E_O_S;
                    result = gTempBuff;


         } ;


      }

       }

////////////////////////////////////
```

```
// affichage rga
if (gIndex > local->Taille_Jacb + local->num_dependant  )
        {

        if ( i_rga < local->num_independant+1 )
        {
           if (j_rga < local->num_dependant+1 )
            {
                        strcpy(gTempBuff, "RGA ( dy");
             strcat(gTempBuff, StringFromIndex(NULL, 0,
                        j_rga++ ));
            strcat(gTempBuff,  "/dx");
                  strcat(gTempBuff, StringFromIndex(NULL, 0,
                       i_rga ));
            strcat(gTempBuff,  " )");
                  if (strlen(gTempBuff) > maxStringLen)
                           *(gTempBuff + maxStringLen) = E_O_S;
                     result = gTempBuff;
           }
           else
           {
             j_rga =1  ; i_rga ++ ;

             strcpy(gTempBuff, "RGA ( dy");
             strcat(gTempBuff, StringFromIndex(NULL, 0,
                        j_rga++ ));
             strcat(gTempBuff,  "/dx");
                  strcat(gTempBuff, StringFromIndex(NULL, 0,
                        i_rga ));
             strcat(gTempBuff,  " )");

                  if (strlen(gTempBuff) > maxStringLen)
                           *(gTempBuff + maxStringLen) = E_O_S;
                     result = gTempBuff;


            }

       }

         }


   if (verbose)
        {
                  if (result)
                           PrintStrings(" = ", result, " -\n");
                  else
                           PrintString(" = NULL -\n");
        }
       return result;


}


/*------------------------------ Initialize ----------------------------*/
Initialize_function
 {
 }       /* Initialize */

                  /*============================================================*/
                  /*                                                            */
                  /*                    Getting Free Variables                  */
                  /*                                                            */
                  /*============================================================*/
```

```
/*------------------------------ PrepareGetOutletFreeInfo ---------------*/
Standard_PrepareGetOutletFreeInfo_function


/*------------------------------ GetNextOutletFreeInfo ------------------*/
Standard_GetNextOutletFreeInfo_function


                    /*=============================================================*/
                    /*                                                             */
                    /*                    Getting and Setting Information          */
                    /*                                                             */
                    /*=============================================================*/

/*------------------------------ GetTypicalValue ------------------------*/
Standard_GetTypicalValue_function

/*------------------------------ GetUnitsForVariable --------------------*/
Standard_GetUnitsForVariable_function


                    /*=============================================================*/
                    /*                                                             */
                    /*                    Calculations                             */
                    /*                                                             */
                    /*=============================================================*/

/*------------------------------ Execute --------------------------------*/
Standard_Execute_function

/*------------------------------ Perform --------------------------------*/
Perform_function
 {
     double calcul_Jacobien;

          RecognizeAllArguments(Perform)
          if (isWarmup)
           {
          // iteration_bump sert comme temps de stabilisation avant de commencer le bump
          // de la variable indépendante suivante car il faut revenir à l'état initial.
          local->iteration_bump = FreeUnitVariable(indexSETTLING_TIME)+1;

          // iteration sert à attendre le temps de stabilisation après lequel on collecte
          // la valeurs des variables dépendantes et on calcul le Jacobien
          local->iteration = 0;

          // total_count est le temps total requis pour faire les bumps les dépump des
          // variables indépendante et le calcul du Jacobien.
          local->total_count = 0;

          // independant_index est l'indice de la calculated variable "CurrentIndependantVar"
          local->independant_index = 0;

          // calculJacobien_index est l'indice de la calculated variable "Jacobien"
          local->calculJacobien_index = 0 ;

          // initialisation à zéro du Jacobien
           for (int j = 0 ; j< (local->num_independant)* (local->num_dependant);j++)
                  {
                   CalcVariable(j +local->num_independant) =  0;
                  }

          // initialisation à zéro de la matrice rga
```

```
            for (int j = 0 ; j< (local->num_independant)* (local->num_dependant);j++)
                  {
                   CalcVariable(j +local->num_independant+local->num_independant + (local->num_independant *
local->num_dependant)) =  0;
                  }

            // initialisation des current variables
             for (int i = 0 ; i< local->num_independant;i++)
               {
                CalcVariable(i) =  FreeUnitVariable(i + 3 );
               }

           if (verbose)
                PrintSelf("- warming up ", " -\n");
       }

     else if (verbose)
                      PrintSelf("- performing ", " -\n");


       if (FreeUnitVariable(indexBUMP) == 1 )
         {

             local->total_count ++ ;
             local->iteration_bump ++ ;
            if ( local->iteration_bump <= FreeUnitVariable(indexSETTLING_TIME))
               goto leave;



            if (  local->total_count < ( 2 * FreeUnitVariable(indexSETTLING_TIME) +1 )* (local->num_independant) +1)
           {

                // bump de la variable indépendante
               if ( local->iteration == 0 && local->iteration_bump > FreeUnitVariable(indexSETTLING_TIME))
                 {
                  CalcVariable(local->independant_index)=FreeUnitVariable(local->independant_index + 3)*
(FreeUnitVariable(indexBUMP_FACTOR)+1);

                 }

                 local->iteration++;

               if (local->iteration > FreeUnitVariable(indexSETTLING_TIME))
                 {
                  //le débump de la calculated variable "CurrentIndependantVar"
                  CalcVariable(local->independant_index)=CalcVariable(local-
>independant_index)/(FreeUnitVariable(indexBUMP_FACTOR)+1);
                  FreeUnitVariable(local->independant_index + 3)= CalcVariable(local->independant_index);
                  for (int i = 0 ; i< local->num_dependant;i++)
                   {
                   // calcul du dx
                   double a =  FreeUnitVariable(local->independant_index + 3)* FreeUnitVariable(indexBUMP_FACTOR)
;

                   // calcul du dy
                   double b =  FreeUnitVariable(local->num_independant + 3 +i) - CalcVariable(i+local->Taille_Jacb) ;

                   //  calcul_Jacobien dy/dx
                   CalcVariable(local->calculJacobien_index + local->num_independant) = b/a;

                   local->calculJacobien_index ++ ;

                   }


                 local->iteration_bump = 0 ;
                 local->iteration = 0;
                 local->independant_index ++;
                 }
```

```
    }

// cette partie du code permet le calcul de la matrice RGA

 else
{

 if (local->num_dependant == local->num_independant )
 {
      // matrice des gains K
     const T_Size  rows = local->num_dependant ;
     const T_Size  columns = local->num_independant;
     const T_Size  elements = local->num_independant * local->num_dependant ;

     // creation du vecteur V contenant les valeur du Jacobien
     int t = 0;
     Pvoid V = CreateVector (elements)  ;
     for (int k = 0 ; k < (local->num_independant * local->num_dependant) ; k++)
     {
     SetVectorValue(V,k,CalcVariable(k + local->num_independant));
     }
     // creation de la matrice des gains K

     Pvoid K = CreateMatrix (rows, columns) ;
     for (int i = 0 ; i < local->num_independant; i++)
     {
        for (int j = 0 ; j< local->num_independant;j++)
        {
         SetMatrixValue(K,j,i,GetVectorValue(V,t));
         t++;
        }
     }
     // inversion de la matrice des gains K avec K_inv

     Pvoid K_inv = CreateMatrix (rows, columns) ;
     // declaration de la transposee de K_inv
     Pvoid K_inv_T = CreateMatrix (rows, columns) ;
     // acquistion de la matrice des gains K
     SetMatrixToMatrix(K_inv,K);
     // inversion de la matrice des gains
     //bool w = IsMatrixSingular(K_inv) ;
     InvertMatrix(K_inv) ;

     // transposee de l'inverse
     SetMatrixToMatrixTransposed(K_inv_T, K_inv);

     // calcul de la matrice rga
     // on calcule la matrice des gains relatif RGA en multipluyant element par element
     // des matrice des gains K et la transposee de l'inverse K_inv_T
     // la multiplication element par element va e faire au niveau de deux vecteurs

     Pvoid K_Vecteur = CreateVector (elements)  ;
     Pvoid K_inv_T_Vecteur = CreateVector (elements)  ;
     int g = 0 ;
     for (int i = 0 ; i < local->num_independant; i++)
     {
        for (int j = 0 ; j< local->num_independant;j++)
        {
         SetVectorValue(K_Vecteur,g,GetMatrixValue(K,j,i));
         SetVectorValue(K_inv_T_Vecteur,g,GetMatrixValue(K_inv_T,j,i));
         g++;
        }
     }
     MultiplyVectorByVector(K_Vecteur,K_inv_T_Vecteur);

     // remplissage matrice rga

     Pvoid RGA = CreateMatrix (rows, columns) ;
```

```
                int u =0;

                for (int i = 0 ; i < local->num_independant; i++)
                {
                    for (int j = 0 ; j< local->num_independant;j++)
                    {
                     SetMatrixValue(RGA,j,i,GetVectorValue(K_Vecteur,u));
                     u++;
                    }
                }

                // mettre les valeurs du rga dans les calculated variables
                // h est l'index calculated variable rga
                int h = local->num_independant+local->num_independant + (local->num_independant * local-
>num_dependant) ;

                for (int i = 0 ; i < local->num_independant; i++)
                {
                    for (int j = 0 ; j< local->num_independant;j++)
                    {
                     CalcVariable(h) = GetMatrixValue(RGA,j,i);
                     h++;
                    }
                }


               DestroyMatrix(RGA);
               FreeUnitVariable(indexBUMP) = 0 ;
               SetFreeUnitVariable(1,0);
               }

              else {
                FreeUnitVariable(indexBUMP) = 0 ;
                SetFreeUnitVariable(1,0);
                 }

              }

        }
          else
           {
           local->iteration = 0;
           local->total_count = 0;
           local->independant_index = 0;
           local->calculJacobien_index = 0 ;

           // initialisation variables calculées
           for (int i = 0 ; i< local->num_independant;i++)
                  {
                   CalcVariable(i) =  FreeUnitVariable(i + 3 );
                  }

           // initialisation de la valeur des variables dépendante à bump =0
           for (int j = 0 ; j< local->num_dependant;j++)
                  {
                   CalcVariable(j+local->Taille_Jacb) = FreeUnitVariable(j + 3 +local->num_independant );
                  }



           }



leave:
          {}
 }        /* Perform */

                    /*============================================================*/
```

```
                    /*                                                                    */
                    /*                          Reading/Writing                           */
                    /*                                                                    */
                    /*====================================================================*/

/*------------------------------ WriteDataToMemory ----------------------*/
Standard_WriteDataToMemory_function

/*------------------------------ ReadDataFromMemory ---------------------*/
Standard_ReadDataFromMemory_function

/*------------------------------ WriteDataToFile ------------------------*/
Standard_WriteDataToFile_function

                    /*====================================================================*/
                    /*                                                                    */
                    /*                          Termination                               */
                    /*                                                                    */
                    /*====================================================================*/

/*------------------------------ Terminate ------------------------------*/
Standard_Terminate_function

/*------------------------------ ShutDown -------------------------------*/
Standard_ShutDown_function
```