Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Supérieure Polytechnique
Département de Génie Electrique
Spécialité : Automatique

PROJET DE FIN D'ETUDES EN VUE DE L'OBTENTION
DU DIPLOME D'INGENIEUR D'ETAT
EN AUTOMATIQUE

Thème

DIFFERENTES TECHNIQUES D'OPTIMISATION DE LA COMMANDE FLOUE. APPLICATION SUR LA COLONNE D'ABSORPTION

Etudié par :

- ATTIA Rachid
- ADJADJI Mohammed

Proposé et dirigé par :

M. ILLOUL Rachid.

2008/2009
Ecole Nationale Supérieure Polytechnique
10, Avenue Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

Remerciements

Nous remercions ***Allah****, Le tout puissant, pour nous avoir donné, le courage, la patience, la volonté et la force nécessaires, pour affronter toutes les difficultés et les obstacles, qui se sont hissés au travers de notre chemin, durant toutes nos années d'études.*

Nous exprimons nos remerciements à notre promoteur monsieur ***Rachid ILLOUL*** *pour l'assistance qu'il nous a témoignée, pour sa disponibilité, pour sa gentillesse, pour ses conseils et orientations sans lesquels ce travail ne verra jamais le jour, qu'il trouve ici l'expression de notre gratitude.*

Nous adressons nos vifs remerciements à monsieur le professeur ***Mohamed TADJINE*** *pour ses conseils et orientations.*

Nous remercions aussi les membres du jury, qui nous ont fait l'honneur de participer à l'évaluation de ce travail.

Nos remerciements les plus sincères sont adressés à nos enseignants, qui ont contribué durant nos études à l'école nationale polytechnique et spécialement les enseignants du département de génie électrique et l'équipe d'analyse mathématiques du département des sciences fondamentales.

Sans oublier, tous ceux qui nous ont aidés de prés ou de loin.

Dédicaces

A mes très chers parents Dada et Yema, avec toute mon affection et toute ma reconnaissance

A mes chers grands-parents Vava et Yema-Louiza

A mon grand-père maternel

A la mémoire de ma grand-mère maternelle

A mes deux frères Chafaa et Coco et à mes deux sœurs Lamia et Mina

A mes oncles Zoubir, Boukhalfa, Khaled, Younes, Mohamed et Boubkeur et à mes tantes Nana-Hassina et Kahina

A mes oncles et tantes maternels

A mes amis (es) et camarades automaticiens, promo 2009

A tous mes amis et amies en particuliers ceux de polytech

A tous ceux qui m'aiment et tous ceux que j'aime en particulier Fifi pour son aide et son soutien

Rachid.

Dédicaces

A celui qui m'a indiqué la bonne voie en me rappelant que la volonté fait toujours les grands hommes .... À mon cher père.

A celle qui a attendu avec impatience les fruits de sa bonne éducation, ... À ma chère mère.

Je remercie mes parents pour leur soutient, leur aide et leurs conseils, sans qui je ne serais pas où j'en suis aujourd'hui.

A ma chère grand-mère « EL HADJA ZAHRA » pour son encouragement tout au long de mon parcours.

A la mémoire de ma tante.

A Mon cher frère Abderazak, et à mes deux chères sœurs Nacera et Ikram.

A mon oncle Abdou et sa petite famille.

A toute la famille ADJADJI.

A mon ami et mon binôme Rachid, et à tous mes amis et amies qui m'ont aidé de prés ou de loin.

Mohammed.

TABLE DES MATIERES

CHAPITRE III : Les Algorithmes génétiques

CHAPITRE IV : Optimisation par essaims particulaires *(PSO)*

CHAPITRE V : Conception du régulateur flou à l'aide d'une technique d'optimisation

Introduction générale

La diversité des problèmes rencontrés en automatique, notamment portant sur la théorie de la commande et de la conception, ont connu une évolution considérable ces dernières années. Parmi ces théories, la commande des systèmes non linéaires à paramètres répartis ne cesse de se perfectionner, en particulier la commande des procédés chimiques tels que les processus de séparation des gaz. En effet l'absorption est une opération très utilisée en génie des procédés, sa modélisation n'est pas une tache facile, en effet, il s'agit d'un procédés très complexe vue la forte non linéarité qu'il présente, et son caractère réparti.

La colonne d'absorption est un processus très utilisé dans l'industrie chimique, il permet le transfert du gaz carbonique d'une phase gazeuse (Air + CO_2) vers une phase liquide, les deux opèrent à contre courant. La monoéthanolamine est ajoutée pour augmenter les capacités d'absorption, et son débit joue le rôle de la variable de commande. Ainsi alors, la colonne d'absorption apparait comme une application très intéressante pour les techniques modernes de commande intelligente.

La modélisation de la colonne d'absorption se base sur l'établissement d'un bilan de matière dans les deux phases liquide et gazeuse, le modèle final est un système d'équation aux dérivées partielles fortement non linéaire, présente un retard pur, sa commande n'est pas facile vue la répartition géométrique de ses paramètres, et aussi la non appariation instantanée de l'effet de l'entrée sur la sortie, mais qu'après un certain temps.

Pour ce qui est de la stratégie de commande, nous avons choisi la commande par la logique floue, en effet, cette nouvelle logique a connu un intérêt important dans la communauté scientifique au cours des dernières années, l'une des raisons de la popularité des régulateurs flous dans la pratique tient au fait qu'il est possible d'incorporer des connaissances linguistiques sur la manière de piloter un processus difficile à modéliser tel que la colonne d'absorption, en effet, leur propriété d'approximation universelle ainsi que leur nature multivariable en ont fait un outil très puissant pour la commande d'un tel système non linéaire complexe.

Le majeur problème des régulateurs flous est qu'il n'existe pas de méthodes systématiques pour la synthèse et le choix de leurs paramètres. Leur conception et synthèse se basent sur de l'expertise et de connaissances a postériori sur le processus à commander, cela fait qu'un fort caractère de subjectivité survient lors de la conception. Pour remédier à ce problème plusieurs techniques ont été développées, parmi ces méthodes on trouve les différents algorithmes d'optimisation, tel que les algorithmes évolutionnaires, appliqués à la recherche de paramètres optimaux d'un contrôleur flou dédié à un problème de commande donné.

L'étude des phénomènes réels est une source d'inspiration en ingénierie informatique, où l'étude et la modélisation des systèmes complexes sont très présentes. Dans le domaine de l'optimisation, de nombreuses méthodes sont inspirées par de telles études. Parmi ces domaines d'inspiration, la biologie est particulièrement prisée par les chercheurs en optimisation. En effet, la modélisation des systèmes « intelligents » rencontrés dans la nature peut servir à créer une « intelligence artificielle », à même de résoudre des problèmes d'optimisation. Un tel exemple d'inspiration est la méthode d'optimisation par algorithmes génétiques qui sont inspirés de la génétique des mécanismes d'évolution de la nature tels que la sélection, le croisement et la mutation génétique. Un deuxième exemple est l'optimisation par la méthode des essaims particulaires (particle swarm optimization) qui est une inspiration directe du comportement des animaux qui vivres en essaims (abeilles…).

Ces algorithmes constituent une approche originale : il ne s'agit pas de trouver une solution analytique exacte, ou une bonne approximation numérique, mais de trouver des solutions satisfaisant au mieux à différents critères, souvent contradictoires. S'ils ne permettent pas de trouver à coup sûr la solution optimale de l'espace de recherche, du moins peut-on constater que les solutions fournies sont généralement meilleures que celles obtenues par des méthodes plus classiques, pour un même temps de calcul.

Dans notre travail nous étudierons alors l'application de ces algorithmes d'optimisation pour la synthèse d'un contrôleur flou afin de pouvoir commander notre colonne d'absorption, et garantir une concentration de référence en CO_2 à la sortie de la colonne, rejetant ainsi les différentes perturbations en concentration et en débit affectant le mélange gazeux à l'entrée en bas de la colonne.

Ce mémoire est organisé en cinq chapitres, succinctement présentés ci-après :

Le premier chapitre donne une description de la colonne d'absorption ainsi que sa modélisation, en exploitant les principes fondamentaux concernant le transfert de matière et les équations du bilan de matière. Dans ce même chapitre on donnera quelques simulations du comportement du système en boucle ouverte et on appliquera une commande classique PI.

Le deuxième chapitre commencera par un rappel sur les notions de base de la logique floue puis l'exploitation de ces notions en commande des processus. Par la suite l'application d'une commande floue sur notre colonne sera présentée et détaillée à travers les différentes simulations.

Dans le troisième chapitre, on présentera en détaille l'optimisation par algorithmes génétiques en explicitant les notions essentielles telles que les opérateurs génétiques de croisement et de mutation, l'opérateur de sélection et le principe de recherche de solutions en faisant évoluer une population d'individus.

Le quatrième chapitre sera consacré à la présentation de la méthode d'optimisation par essaims particulaires (*Particle Swarm Optimization*) comme étant un outil récent d'optimisation inspiré du comportement psycho-sociale des animaux et insectes vivant en groupes. Nous présenterons une version classique de l'algorithme ainsi que ses variantes adaptatives.

Dans le dernier chapitre on appliquera les deux méthodes d'optimisation pour la synthèse d'un régulateur flou pour la commande de notre colonne d'absorption, ainsi, on présentera les résultats de l'optimisation en les comparants avec la première commande floue non optimisée donnée dans le deuxième chapitre. Puis on essaiera de comparer les résultats issus des deux méthodes d'optimisation.

Enfin, une conclusion générale résumant et évaluant les résultats trouvés dans le cadre de ce mémoire sera donnée. Elle inclura, aussi, divers axes de perspectives envisagées.

Chapitre I :

Modélisation et Simulation de la Colonne d'Absorption

Nomenclature:

A, B	CO_2, MEA
R	-CH_2-CH_2-OH
a	Aire interfaciale par unité de volume du garnissage ($m^2.m^{-3}$)
C_{Ag}, C_{AL}	Concentration du CO_2 dans la phase gazeuse et dans la phase liquide (mol/m^3)
$C_{BL}(x)$	Concentration du composé B dans la phase liquide (mol/m^3)
C_{ge}, C_{Be}	Concentration du CO_2 dans la phase gazeuse et de la MEA dans la phase liquide à l'entrée de la colonne ($mol.m^{-3}$).
C_{gs}, C_{Bs}	Concentration du CO_2 dans la phase gazeuse et de la MEA dans la phase liquide à la sortie de la colonne ($mol.m^{-3}$).
C_{Bli}	Concentration de la MEA dans la phase liquide à l'interface gaz-liquide
C_{Ali}, C_{Agi}	Concentration du CO_2 dans la phase liquide et dans la phase gazeuse à l'interface gaz-liquide ($mol.m^{-3}$).
G, L	Débit volumique de la phase gazeuse et liquide ($m^{-3}.s^{-1}$)
U_g, U_l	Vitesse d'écoulement du gaz et du liquide ($m.s^{-1}$)
F_A	Le flux du composé CO_2 transféré de la phase gazeuse vers la phase liquide ($mol.m^{-3}.s^{-1}$)
N_A	Flux de matière du composé A transféré vers la phase liquide dans le cas du transfert de matière avec réaction chimique (mol/m^2 .s)
N'_A	Flux sans réaction chimique (mol/m^2 .s)
k_L	Coefficient de transfert de matière dans la phase liquide (m/s)
k	Constante de vitesse de réaction $(m^3/mol.s)$
k_A	Coefficient de transfert de matière du composé A dans la phase liquide, pour le cas de transfert de matière sans réaction chimique (m/s)
k_{AG}	Coefficient de transfert de matière du composé A dans la phase gazeuse (m/s)
k_{AL}	Coefficient de transfert de matière du composé A dans la phase liquide (m/s)
D_{Al}	Coefficient de diffusion du CO_2 dans la phase liquide ($m^2.s^{-1}$)
D_{BL}	Coefficient de diffusion du composé B dans la phase liquide (m^2/s)
r_A	Vitesse de la réaction $(mol/m^3.s)$
Ha	Nombre adimensionnel appelé nombre de Hatta
J	Nombre adimensionnel
E	Facteur d'accélération
S	Section de la colonne (m^2)
dz	Hauteur d'une tranche de la colonne d'absorption

Sur les systèmes aux paramètres distribués :

Les SPD concernent un large spectre d'importantes applications industrielles. En particulier dans le domaine du génie des procédés, notamment chimique et biochimique, biotechnologie, thermique,…et d'autres. Enfin pour la quasi-totalité des systèmes physique le caractère distribué est bien présent. En effet tous les paramètres physiques sont fonction d'une distribution spatio-temporelle car ils ne sont pas uniformes suivant l'espace géométrique, et par conséquent leurs caractéristiques ne peuvent être considérées indépendantes de la variable d'espace. L'hypothèse de l'homogénéité se restreint à des cas très particulier.

Par exemple, la température dans une enceinte thermique n'est pas la même en tout point de l'espace mais elle est distribuée, ou encore la conductivité thermique d'un matériau hétérogène constitue un paramètre distribué. Il en est de même pour les concentrations de produits dans un réacteur chimique, le déplacement d'une structure flexible, le débit et la densité d'un écoulement, l'évolution d'une épidémie dans une population qui sont toutes des variables de nature distribuée et ne peuvent pas être ignorées dans un problème de commande.

Dans la littérature, on rencontre de nombreux exemples d'applications industrielles essentielles décrites par des équations aux dérivées partielles. On peut citer les réacteurs (bio) chimiques, laminoirs, cristallisoirs, réacteurs thermiques, échangeurs de chaleur **[35]**, colonne chromatographique, colonne de distillation ou d'absorption à garnissage **[44]**, bras manipulateurs flexibles **[24]**. D'un point de vu représentation d'état, ces systèmes sont de dimension infinie, et les outils classiques en automatique utilisé pour les systèmes à paramètres localisés, c'est-à-dire des systèmes décrits par des équations différentielles ordinaire ODE de dimension finie, ne sont donc plus utilisable directement pour des systèmes physiques à variables caractéristiques non homogènes géométriquement.

L'étude des SPD a connu un regain d'intérêt de la part de la communauté scientifique lors de ces dernières années. Soutenue par un développement parallèle de l'informatique et des outils mathématiques, de plus en plus élaborés, plusieurs travaux de recherche ont été menés et axés sur la modélisation, la simulation, l'identification, l'estimation, le placement optimal des capteurs, et la commande. Les recherches menées ont conduits à de probants résultats sur le plan théorique et pratique.

De nombreux articles de synthèse ont été consacrés à la commande des PDS **[45]**.

Dans notre travail on s'intéresse à la commande d'une colonne d'absorption à garnissage qui est un système aux paramètres répartis, fortement non linéaire. Les paragraphes suivants traitent la modélisation et la simulation de notre système.

I. Modélisation de la colonne d'absorption :

1. Introduction :

L'absorption est une opération unitaire de génie des procédés caractérisée par des transferts de matière d'une phase à une autre. Parfois ces transferts de matière sont accompagnés de transfert de chaleur. L'absorption met en jeu des échanges de matière entre une phase gazeuse et une phase liquide de natures chimiques différentes. Un ou plusieurs constituants de la phase gazeuse passent en solution. Cette opération est principalement utilisée pour purifier un flux gazeux ou pour récupérer un constituant présent dans un mélange gazeux.

Les absorbeurs ont tous pour but de réaliser le meilleur échange de matière entre une phase liquide et une phase gaz en contact. Ils doivent donc être équipés de dispositifs internes qui, d'une part, favorisent la dispersion de la phase gaz dans la phase liquide et plus particulièrement provoquent la plus grande surface d'aire interfaciale, et d'autre part, permettent la séparation de la phase gaz et de la phase vapeur en contact afin d'en faciliter l'écoulement global.

Les performances globales de l'absorbeur, rendement et sélectivité, dépendent des phénomènes mis en jeu, à savoir :

- les équilibres thermodynamiques à l'interface (solubilités) ;
- les lois de transport dans les phases (diffusivités) ;
- les lois de transfert au voisinage des interfaces (coefficients de transfert, aires interfaciales) ;
- les cinétiques des réactions chimiques (schémas réactionnels, constantes cinétiques, ordres de réactions).

Les colonnes d'absorption du **CO_2** par des solutions aqueuses de monoéthanolamine (**MEA**) sont largement utilisées comme unités de séparation dans l'industrie chimique. Les principales utilisations sont dans l'industrie de fabrication de l'ammoniac et dans les unités de traitement du gaz naturel du fait de leur simplicité de conception et d'utilisation.

Pour la commande de ces processus, on a recours dans la majeure partie des cas à un modèle de conduite de type boite noire , mais pour améliorer les performances et aboutir à un modèle mathématique du fonctionnement de la colonne nous sommes amenés à utiliser un modèle de connaissance de type génie chimique qui permet d'établir un modèle du procédé en se basant sur les lois fondamentales de la physique et de la chimie, telles que la conservation de masse, les équilibres entre phases, et les lois de transfert.**[3]**

En établissant les bilans de matière dans les deux phases liquide et gazeuse, on développera un modèle mathématique décrivant le fonctionnement en régime dynamique

de la colonne d'absorption à garnissage opérant à contre courant, le modèle obtenu est un modèle fortement non linéaire à paramètres répartis (distribués).

2. Description et principe de fonctionnement :

Une colonne est une unité de séparation physico-chimique utilisée en chimie et en biologie. Il s'agit en général d'un tube dans lequel passe un ou plusieurs mélanges et qui permet de séparer un ou plusieurs composés du mélange principal. Le principe de séparation est variable et utilise différents moyens. On désigne ces unités en fonction de leur principe de séparation, ex. colonne d'absorption, colonne de distillation, colonne de chromatographie,...Les modes de fonctionnement des colonnes peuvent être continu ou discontinu (batch).

Dans notre cas, la colonne d'absorption utilisée est une colonne en garnissage en verre, mesurant ***1,26 m*** de hauteur et ***75 mm*** de diamètre intérieur. La colonne possède deux tronçons superposés dont la hauteur est de ***63 cm*** chacun et dont le but est de recentrer le liquide et de permettre la prise d'échantillons liquides et gazeux ***(Figure 1.1)***, chaque tronçon contient des grains de garnissage. Le choix du garnissage, qui est un élément essentiel, est dicté par la surface de contact offerte entre le gaz et le liquide utilisé, le calcul des pertes de charge et son prix. Les garnissages peuvent être de formes variées (anneaux, selles…), de matériaux différents (céramique, verre, métal…) et être rangés ou disposés en vrac. La colonne fonctionne en circuit ouvert ou fermé. Dans notre étude, le garnissage est disposé en vrac et est du type anneaux de **Raschig**, de dimension caractéristique de ***10mm*** destiné à améliorer la surface de contact entre phases.

Figure 1.1. *La colonne d'absorption*.

La pression et la température de travail sont respectivement de ***1,2 bar*** et de ***25°C***. L'amplitude des perturbations en composition du CO_2 dans le mélange à traiter est de l'ordre de ***10*** à ***20%.*** Cette colonne est utilisée pour réduire la concentration du CO_2 à une valeur désirée **[1] [3]**.

Le processus d'absorption étudié dans ce travail consiste à retirer d'un mélange gazeux (***air+CO_2***) le gaz acide (CO_2) qu'il contient, et cela en utilisant un liquide de lavage (***monoéthanolamine : MEA+eau***). Les deux phases (gaz, liquide) opèrent à contre courant.

Le principe de fonctionnement du processus est le suivant : Du gaz chargé de CO_2 circule dans la colonne du bas vers le haut (ascendant). Une solution aqueuse de liquide de lavage, ici la ***MEA*** circule à contre-courant, par gravité sur le garnissage. Lors du contact entre phase liquide et gazeuse sur la surface des anneaux de ***Raschig***, le CO_2 passe de la phase gazeuse vers la phase liquide ; cette diffusion est accélérée par réaction chimique du CO_2 avec la ***MEA*** dans la phase liquide. La ***MEA*** est ainsi considéré comme un absorbant. Ce liquide va se charger en gaz carbonique et nous aurons donc en tête de la colonne le gaz épuré ***(Figure 1.2).***

Par mesure d'économie, la ***MEA*** est régénérée (épurée du CO_2 quelle contient) par élévation de température (***110°C***) puis recyclée dans le pilote. Le débit de ***MEA*** et la concentration du CO_2 dans le mélange gazeux sont respectivement sélectionnés comme variables de commande et de sortie.

La modélisation permet d'établir un modèle mathématique à partir de lois fondamentales telles que la conservation de la masse, le premier principe de la thermodynamique, les équilibres entre phases et les lois de transfert.

Nous présenterons dans cette partie les différentes étapes conduisons à l'établissement d'un modèle mathématique, représentant la dynamique de la colonne d'absorption du CO_2 d'un mélange gazeux (***CO_2+air***) par une solution aqueuse de Monoéthanolamine (***MEA***) **[5]**.

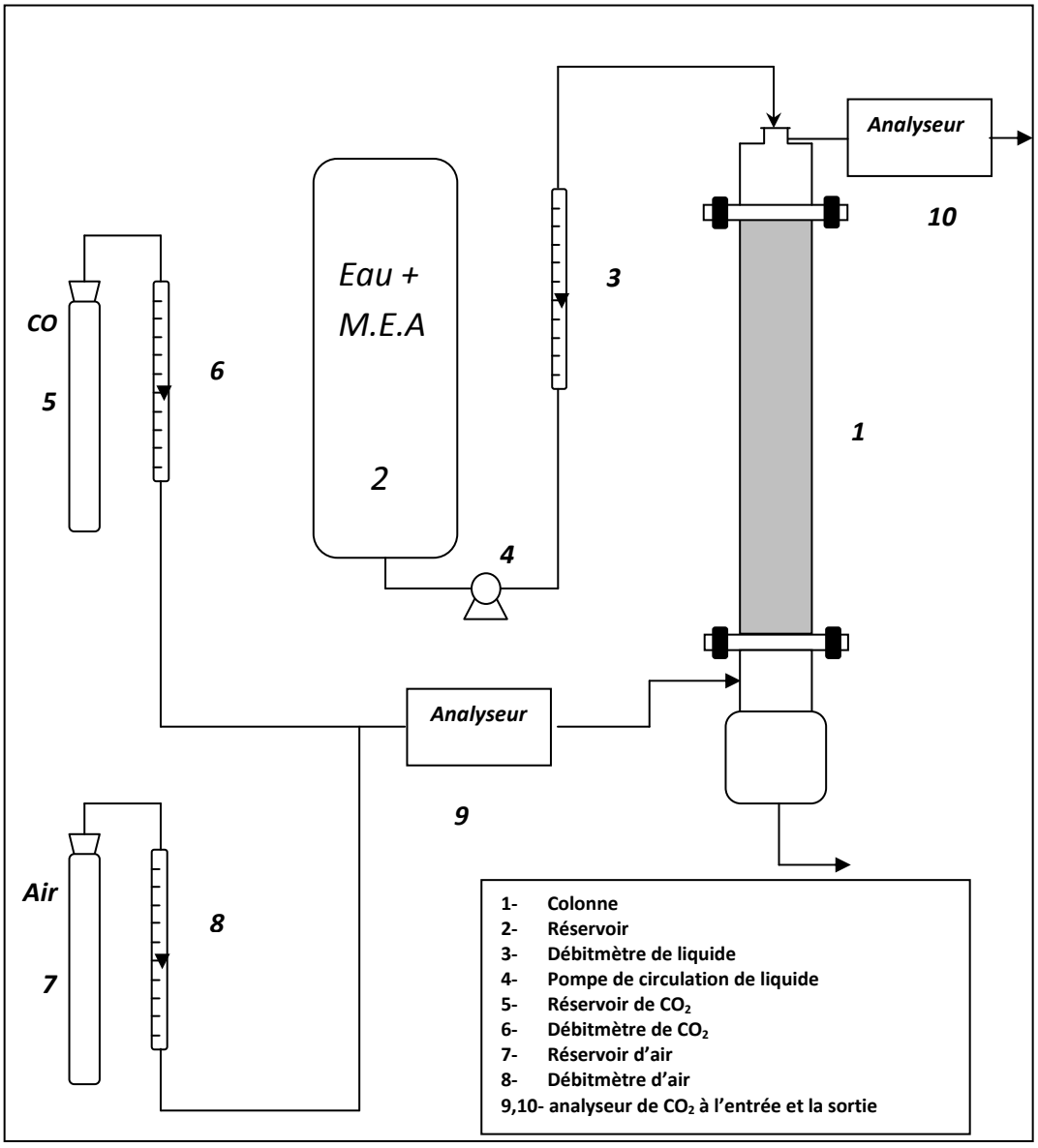


Figure 1.2. Schéma de la colonne d'absorption

3. Transfert de matière avec réaction chimique

3.1. Principe

L'absorption d'un gaz accompagnée d'une réaction chimique dans la phase liquide permettra d'améliorer l'efficacité de séparation (par exemple l'absorption du CO_2 dans une solution aqueuse de soude. amine ...etc.), et la sélectivité de séparation (par exemple l'absorption du CO_2 et H_2S dans des solutions d'éthanolamine).

Une absorption avec réaction chimique va accélérer le transfert du soluté de la phase gazeuse vers la phase liquide.

La réaction à étudier étant la suivante :

$$\underset{\underbrace{I + A}_{phase\ gazeuse}}{(Inerte + Soluté)} + \underset{\underbrace{S + B}_{phase\ liquide}}{(Solvant + Réactif)} \rightarrow \underset{\underbrace{S,B,P}_{phase\ liquide}}{(\mathrm{Pr}oduit)} + \underset{\underset{\sim}{I}\ phase\ gazeuse}{(Inerte)}$$

$$air + CO_2 \qquad eau + MEA$$

Au cours de laquelle le soluté (A) réagit avec le réactif (B) en solution pour donner le produit (P) d'après les réactions suivantes :

$A + q\,B \rightarrow p\,P$ *réaction totale*

$A + q\,B \leftrightarrow p\,P$ *réaction équilibrée*

Comme toute opération de transfert de matière, l'absorption avec réaction chimique peut être décomposée en plusieurs étapes :

- Transfert de A de la phase gazeuse vers l'interface gaz/liquide ;
- Transfert de A de l'interface vers la phase liquide ;
- Transfert du réactif dans la phase liquide vers le site réactionnel ;
- Réaction entre A et B ;
- Transfert de P du site réactionnel vers la phase liquide **[1]**.

Cette réaction a un double effet :

• L'augmentation de la capacité d'absorption du liquide puisque le réactif fait disparaître le soluté absorbé et la force motrice du transfert est donc accrue.

• L'accroissement de la vitesse avec laquelle le soluté franchit l'interface pour passer dans le liquide, le coefficient de transfert de matière augmente par conséquent.

Ce dernier effet est pris en compte dans les expressions du flux spécifique d'absorption en multipliant le coefficient de transfert de matière en phase liquide sans réaction chimique par un facteur appelé facteur d'accélération, noté **E**.

Le flux spécifique d'absorption avec réaction chimique s'écrira donc :

$$Fa = N_A.a = E.k_L.a.(C_{ALi} - C_{AL}) \qquad (1.1)$$

Avec :

Fa : Le flux volumique d'absorption du composé A $(mol/m^3.S)$

N_A : Flux surfacique du composé A, pour les cas du transfert de matière avec réaction chimique $(mol/m^2.S)$

a : Aire interfaciale (m^2/m^3)

C_{ALi} : Concentration du composé A à l'interface coté liquide (mol/m^3)

C_{AL}: Concentration du composé A dans la phase liquide $(mol\ /m^3)$

k_L: Coefficient de transfert de matière dans la phase liquide (m/s)

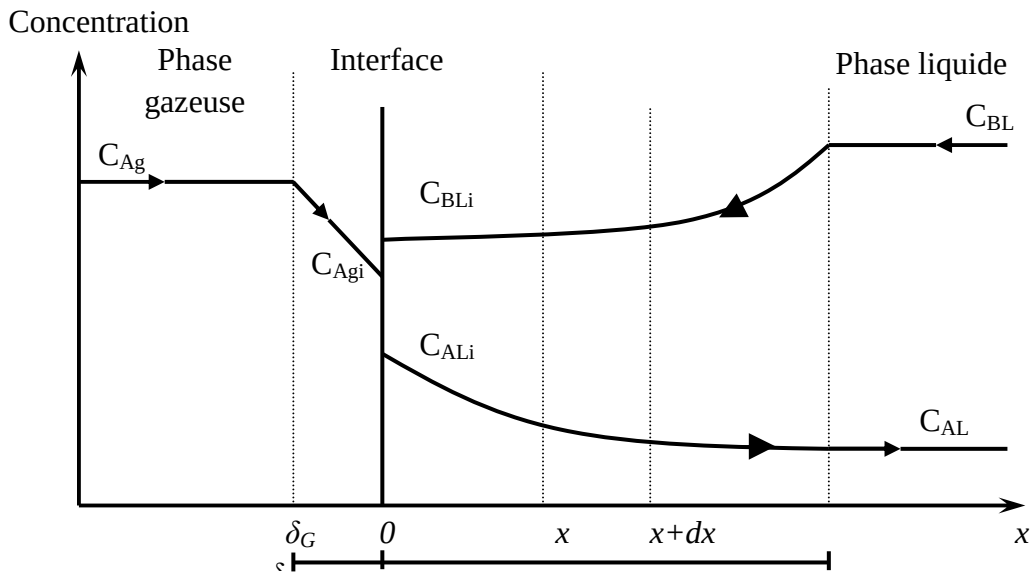


Figure 1.3. Profils des concentrations pour le transfert de matière avec réaction chimique dans le sens transversal

3.2. Bilan de matière dans le sens longitudinal

Pour simplifier le traitement numérique et alléger la structure de notre modèle, on adopte les hypothèses suivantes :

- Il n y a pas de résistance en phase gazeuse.
- Le processus est isotherme.
- Réaction rapide entre le CO_2 et la MEA $(Ha\ >\ 5)$.
- La dispersion axiale est négligeable dans la phase gazeuse et dans la phase liquide.

Dans ces conditions, les équations du modèle se réduisent à l'écriture des bilans de matières partiels dans chaque phase, auxquelles s'ajoutent les relations traduisant les conditions aux limites.

Puisque, on s'intéresse à l'évolution de la concentration du CO_2 le long de la colonne d'absorption, on effectue le bilan de matière en soluté CO_2 sur la phase gazeuse et sur un élément dz ***(Figure 1.4)*** **[1]**.

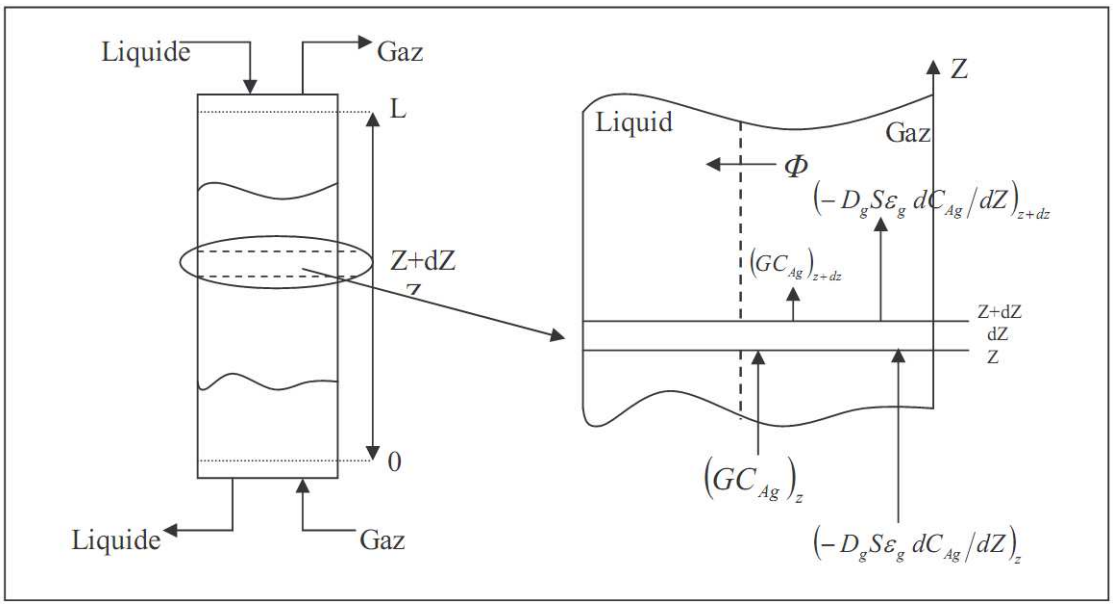


Figure 1.4. Bilan de matière sur une tranche élémentaire dz (dans le sens longitudinal)

❖ Le bilan de matière pour le CO_2 dans la phase gazeuse pour une tranche élémentaire de hauteur dz et de largeur (z+dz) s'écrit :

Quantité de soluté à l'entrée z	=	Quantité de soluté à sortie z+dz	+	Quantité de soluté transféré de la phase gazeuse vers la phase liquide	+	Accumulation dans l'élément dz

Ceci donne :

$$(GC_{Ag})_z = (GC_{Ag})_{z+dz} + F_a S\, dz + S\frac{dC_{Ag}}{dt}dz \qquad (1.2)$$

Or :

$$G \cdot C_{Ag}|_{z+dz} = G \cdot C_{Ag}|_z + \frac{d}{dz}(G \cdot C_{Ag})dz \qquad (1.3)$$

On remplace dans (1.2), on obtient :

$$U_g \frac{dC_{Ag}}{dz} + F_A = -\frac{dC_{Ag}}{dt} \qquad (1.4)$$

$U_g = G/S$ (m/s) étant la vitesse moyenne d'écoulement.

❖ Dans la phase liquide, le CO_2 réagit avec le monoéthanolamine (MEA) suivant la formule : **[2]**

$$CO_2 + 2\,RNH_2 \leftrightarrow RNHCOO^- + RNH_3^+ \qquad (1.5)$$

La vitesse r_A de cette réaction $(mol/m^3.s)$ a la forme suivante: $\boldsymbol{r_A} = k\, C_{Al} C_{Bl}$

$$\text{Avec : } Log_{10}k = 10{,}99 - \frac{2152}{T}\ (l/mole.s) \qquad (1.6)$$

T : Température (K)

k : Constante de la vitesse de la réaction $(l/mole.s)$

En tenant compte de l'expression de r_A, le bilan de matière pour le CO_2 dans la phase liquide donne:

Quantité de soluté à l'entrée z	=	Quantité de soluté à sortie z+dz	+	Quantité de soluté transféré de la phase gazeuse à la phase liquide	+	Quantité de soluté apparaissant ou disparaissant par unité de temps	+	Accumulation

$$(LC_{Al})_z = (LC_{Al})_{z+dz} + F_A\, S\, dz - [k\, C_{Al}\, C_{Bl}]\, S\, dz + S\frac{dC_{AL}}{dt}\, dz \qquad (1.7)$$

Or:
$$(LC_{Al})_{z+dz} = (LC_{Al})_z + \frac{d}{dz}(LC_{Al})dz \qquad (1.8)$$

$$\frac{d}{dz}(LC_{Al})_z + F_A.S.dz - (k.C_{AL}.C_{BL}).S.dz + S\frac{dC_{AL}}{dt}dz = 0 \qquad (1.9)$$

D'où:

$$U_l\ \frac{dC_{Al}}{dz} + F_A - [k\ C_{Al}\ C_{Bl}] + \frac{dC_{AL}}{dt} = 0 \qquad (1.10)$$

$U_l = L/S\ \ (m/s)$ étant la vitesse moyenne d'écoulement du flux liquide.

D'une manière générale, la réaction entre le CO_2 et le MEA est considérée comme une réaction rapide (nombre de Hatta > 5) **[2]**, ce qui implique qu'il n'y a pas d'accumulation du CO_2 dans la phase liquide et pas de variation en fonction du temps, on peut donc écrire :

$$\begin{cases} \dfrac{dC_{Al}}{dz} = 0 \\ \dfrac{dC_{Al}}{dt} = 0 \end{cases} \Rightarrow F_A = [k\ C_{Al}\ C_{Bl}] \qquad (1.11)$$

Ceci veut dire que la quantité du CO_2 transférée dans la phase liquide réagit totalement avec la MEA.

- Le bilan de matière sur la MEA dans la phase liquide, en tenant compte du coefficient stœchiométrique de la réaction (1.5) et du fait que la MEA ne peut pas passer dans la phase gazeuse, donne :

$$(LC_{Bl})_z = (LC_{Bl})_{z+dz} - 2[k\, C_{Al}\, C_{Bl}]\; S\, dz - S\frac{dC_{Bl}}{dt}\, dz \tag{1.12}$$

$$(LC_{BL})_{z+dz} = (LC_{BL})_z + \frac{d}{dz}(LC_{BL})dz \tag{1.13}$$

$$\frac{d}{dz}(LC_{Bl})_z - 2(k.C_{AL}.C_{BL}).S.dz - S\frac{dC_{BL}}{dt}dz = 0 \tag{1.14}$$

D' où:

$$U_l\, \frac{dC_{Bl}}{dz} - 2\,[k\; C_{Al}\; C_{Bl}] = \frac{dC_{BL}}{dt} \tag{1.15}$$

En tenant compte de (1.11), on obtient :

$$U_l \frac{dC_{Bl}}{dz} - 2\, F_A = \frac{dC_{Bl}}{dt} \tag{1.16}$$

Les bilans de matière sur le CO_2 et la MEA dans les phases liquide et gazeuse se réduisent aux équations suivantes :

- Bilan sur le CO_2 dans la phase gazeuse :

$$U_g\; \frac{dC_{Ag}}{dz} + F_A = -\frac{dC_{Ag}}{dt} \tag{1.17}$$

- Bilan de la MEA dans la phase liquide :

$$U_l \frac{dC_{Bl}}{dz} - 2\, F_A = \frac{dC_{Bl}}{dt} \tag{1.18}$$

- Expression de F_A :

$$F_A = [Na]\, a \tag{1.19}$$

$$[Na] = E.[N'a] = E.k_{AL}.(C_{ALi} - C_{AL}) \quad \text{Avec} \quad C_{AL} \approx 0 \quad \text{(réaction rapide)} \tag{1.20}$$

Donc :

$$F_A = a\, E\, k_{Al}\, C_{ALi} \tag{1.21}$$

En admettant que l'équilibre est établi à l'interface gaz - liquide et pour des faibles concentrations du CO_2, nous pouvons écrire :

$$C_{AGi} = m.C_{ALi} \tag{1.22}$$

$$[N_A] = k_{AG}(C_{AG} - m.C_{ALi}) = E.k_{AL}.C_{ALi} \tag{1.23}$$

D' où: $$C_{Ali} = \frac{K_{Ag}\ C_{Ag}}{E\ K_{Al} + m\ K_{Ag}} \tag{1.24}$$

Il faut finalement tenir compte des conditions aux limites qui sont pour le gaz la concentration du CO_2 en bas de la colonne ou concentration d'entrée $\boldsymbol{C_{Age}}$ et pour le liquide la concentration de la MEA en haut de la colonne ou concentration d'entrée $\boldsymbol{C_{Ble}}$.

Notre colonne d'absorption est finalement décrite par le système d'équations aux dérivées partielles suivant:

$$\begin{cases} U_g \frac{dC_{Ag}}{dz} + F_A = -\frac{dC_{Ag}}{dt} \\ U_l \frac{dC_{Bl}}{dz} - 2F_A = \frac{dC_{Bl}}{dt} \end{cases} \tag{1.25.a}$$

Avec les conditions aux limites suivantes:

$$\begin{cases} C_{Ag}\big|_{z=0} = C_{Age} \\ C_{Bl}|_{z=h} = C_{Ble} \end{cases} et \begin{cases} \frac{\partial C_{Ag}}{\partial z}\Big|_{z=L} = 0 \\ \frac{\partial C_{Bl}}{\partial z}\Big|_{z=0} = 0 \end{cases} \tag{1.25.b}$$

Les valeurs des différents paramètres utilisés lors de la simulation sont données dans le tableau 1.1 suivant :

Paramètre	**Valeur / expression**
S	***0,0044 m²***
K	***5,9 $m^3.mol^{-1}.s^{-1}$***
D_{Al}	***1,51.10^{-9} $m^2.s^{-1}$***
D_{Bl}	***1,1.10^{-9} $m^2.s^{-1}$***
K_{Ag}	***5,81. 10^{-2} $(U_g)^{0,7}$***
K_{Al}	***2.10^{-2} $(U_l)^{0,67}$***
A	***469,1[1-exp(-3,3.$U_l^{0,4}$)]***
M	***1,18***
Log_{10} (k)	***10,99 – 2152 /T***

Tableau 1.1 *: Paramètres du modèle pour* ***T=25°C***

4. Conclusion :

Dans ce paragraphe, nous avons donné une description d'une colonne d'absorption opérant à contre courant, ainsi que son mode et son principe de fonctionnement, puis , en utilisant les principes fondamentaux concernant le transfert de matière avec réaction chimique ainsi que les bilans de matière dans les deux phases : liquide et gazeuse, nous avons développé le modèle mathématique décrivant notre processus d'absorption du CO_2 par le Monoéthanolamine (MEA).

II. Simulation en boucle ouvert et régulation PI :

Le modèle obtenu est un système aux paramètres répartis et sa simulation n'est pas aussi aisée que pour les modèles aux équations différentielles ordinaires, car il est équivalent à un système d'état d'ordre infini. Parmi les méthodes les plus répandues pour la simulation des PDE on trouve, la méthode des différences finies, la méthode des éléments ou des volumes finis. L'établissement d'un algorithme de simulation passe par plusieurs phases dont le choix des pas de discrétisation spatiale et temporelle et l'établissement d'un schéma numérique stable.

L'idée principale de ces méthodes est le remplacement des dérivées spatiales et temporelles par une différentiation finie. Ainsi on remplace le problème en PDE au départ continu par un problème discret à un nombre d'inconnues fini. On obtient généralement un système d'équations algébriques non linéaire de grande dimension, puis on résout ce système d'équation en appliquant un algorithme de résolution des systèmes non linéaire, on trouve dans la littérature plusieurs algorithmes développés pour cette fin. L'algorithme de résolution est bien évidement un algorithme itératif initialisé en exploitant les conditions initiales ainsi que les conditions aux limites, c'est pour cette raison ci qu'un profil initiale, c'est-à-dire à $t = 0$ est indispensable.

Pour notre problème nous avons appliquées l'outil disponible sous Matlab 7.1® pour la résolution des équations aux dérivées partielles, qui est la routine ***pdepe***.

1. Description de la fonction *pdepe* :

La fonction ***pdepe*** est conçue pour la résolution des équations aux dérivées partielles paraboliques et elliptiques unidimensionnelles.

La forme générale de l'équation est la suivante :

$$c\left(x,t,u,\frac{\partial u}{\partial x}\right)\frac{\partial u}{\partial x}=x^{-m}\frac{\partial}{\partial x}\left(x^{m}f\left(x,t,u,\frac{\partial u}{\partial x}\right)\right)+s\left(x,t,u,\frac{\partial u}{\partial x}\right)$$

Syntaxe :

$$sol = pdepe(m, pdefun, icfun, bcfun, xmesh, tspan)$$

Où :

- m : Correspond aux coordonnées cartésiennes (m=0), cylindriques (m=1) ou sphériques (m=2).
- $xmesh$: Défini le maillage dans l'espace
- $tspan$: Défini le maillage dans le temps
- $pdefun$: l'équation aux dérivées partielles en elle même est définie par les trois fonctions $c, f, et\ s$ de la formule générale **[9]**.

$$[c, f, s] = pdefun(x, t, u, dudx)$$

- $icfun$: la fonction définissant la condition initiale.
- $bcfun$: la fonction définissant les conditions aux limites.

2. Simulation en boucle ouverte :

Après mise en forme du modèle obtenu de la colonne selon la fonction ***pdepe***, et un choix adéquat du maillage de l'espace et du pas de discrétisation temporelle, les réponses en boucle ouverte pour différentes perturbations sont représentées par les figures ci-dessous.

- **Point de fonctionnement :**

La concentration de CO_2 en entrée est de $5.79\ mole/m^3$.

La valeur obtenue en régime permanent est de $3.975 mole/m^3$.

Débit de gaz nominal : $1.9255\ l/s$.

Débit de liquide nominal : $0.0213\ l/s$.

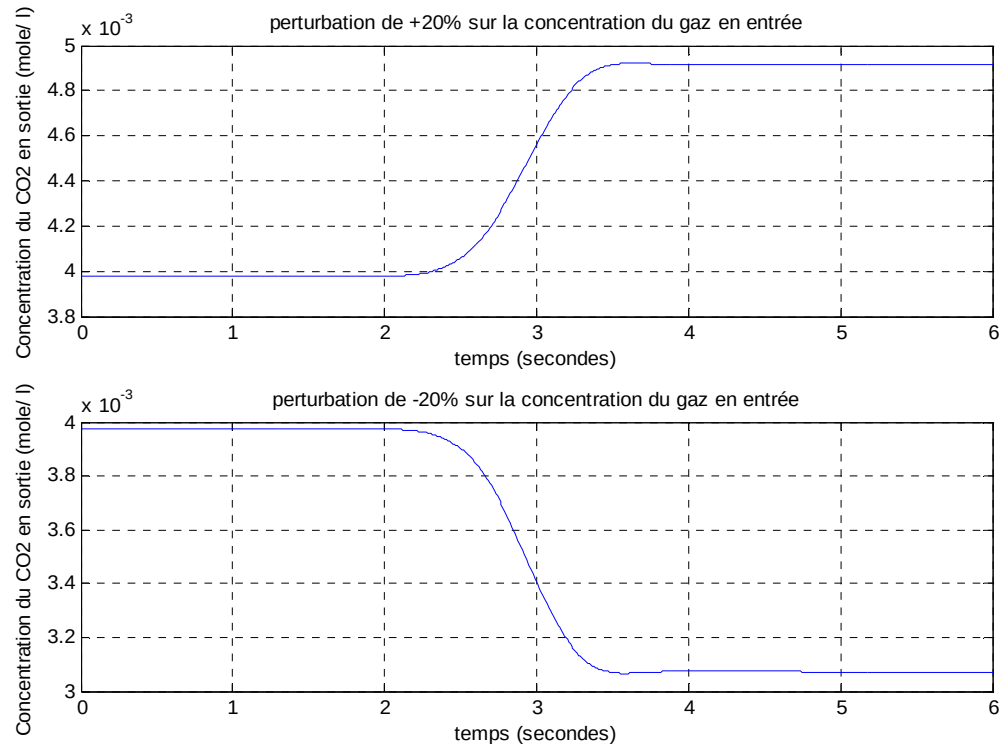


***Figure 1.5** : Perturbation de $\pm 20\%$ sur la concentration du gaz en entrée.*

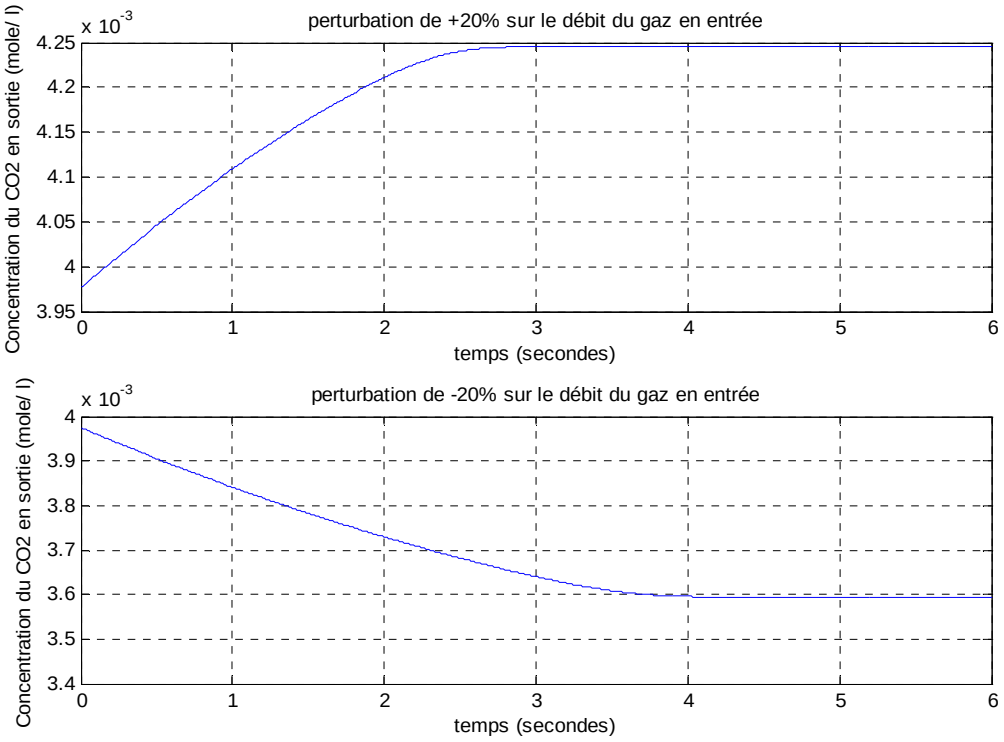


***Figure 1.6**. Perturbation de $\pm 20\%$ sur le débit du gaz en entrée.*

3. Régulation PI :

Nous avons appliqué une Régulation PI au modèle dynamique de notre colonne d'absorption développé dans la partie modélisation. Pour faciliter la simulation de la colonne d'absorption, nous avons choisi une commande échantillonnée. La période d'échantillonnage choisie est égale à 0.5 seconde, la teneur régulée moyenne en sortie est de $3.975\ 10^{-3}$ mole de CO_2 par litre de gaz.

Tout au long de ce mémoire les tests sur les régulateurs seront performés pour les perturbations suivantes :

- $+30\%$ sur la concentration du gaz en entrée.
- -20% sur la concentration du gaz en entrée.
- $\pm\ 30\%$ sur le débit du gaz en entrée.

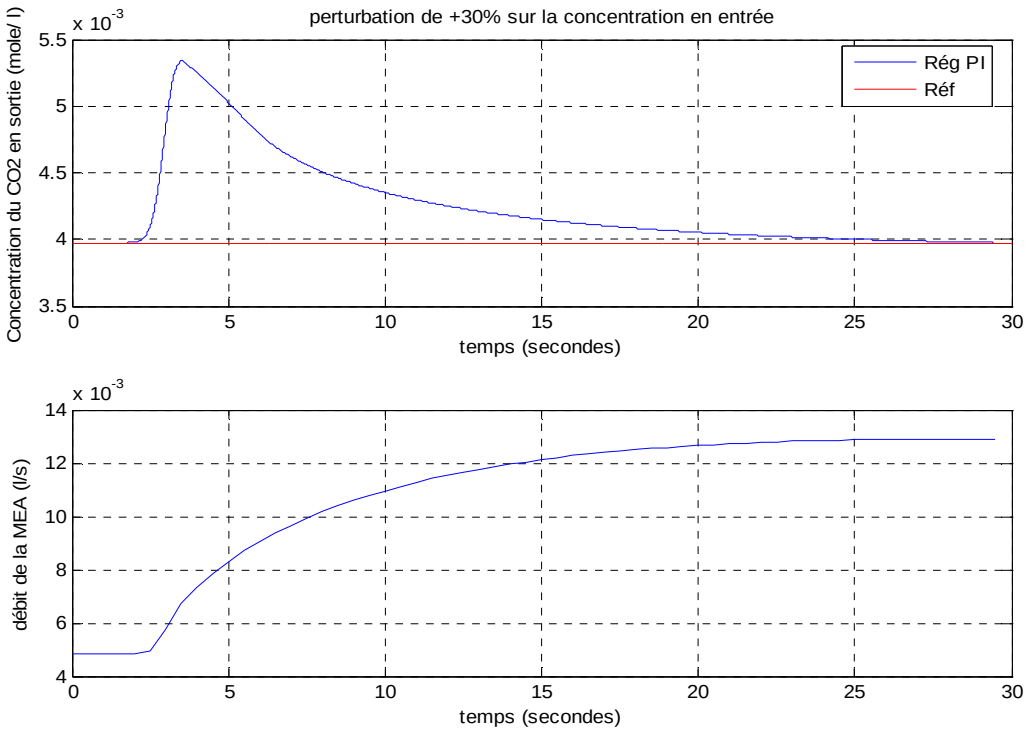


Figure 1.7. *Réponse pour une perturbation de +30% sur la concentration du gaz en entrée.*

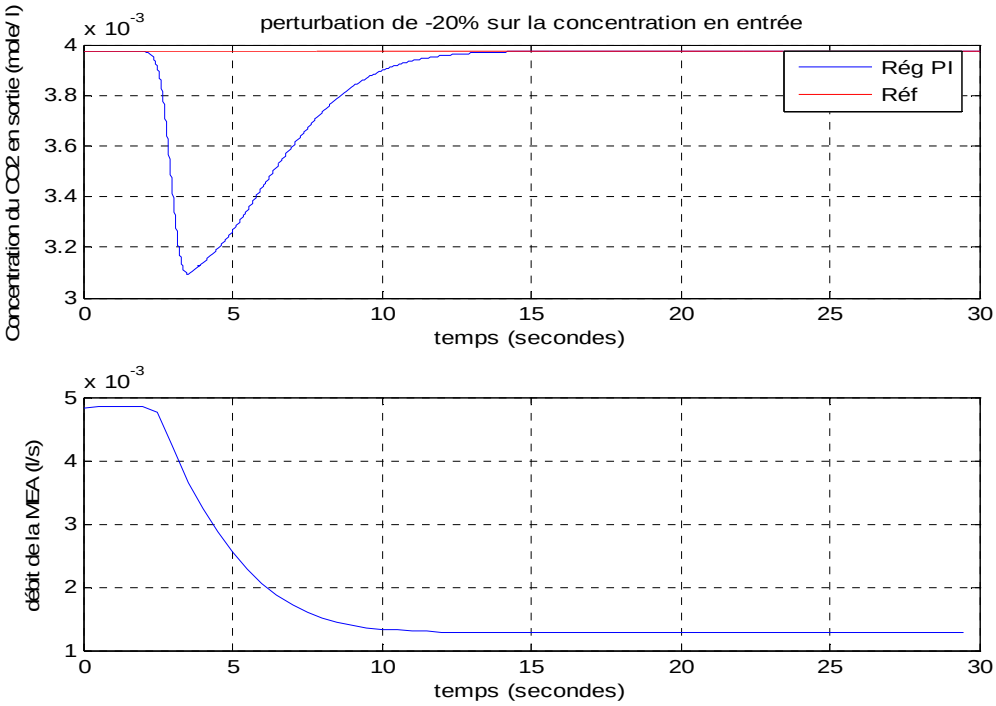


Figure 1.8. *Réponse pour une perturbation de -20% sur la concentration du gaz en entrée.*

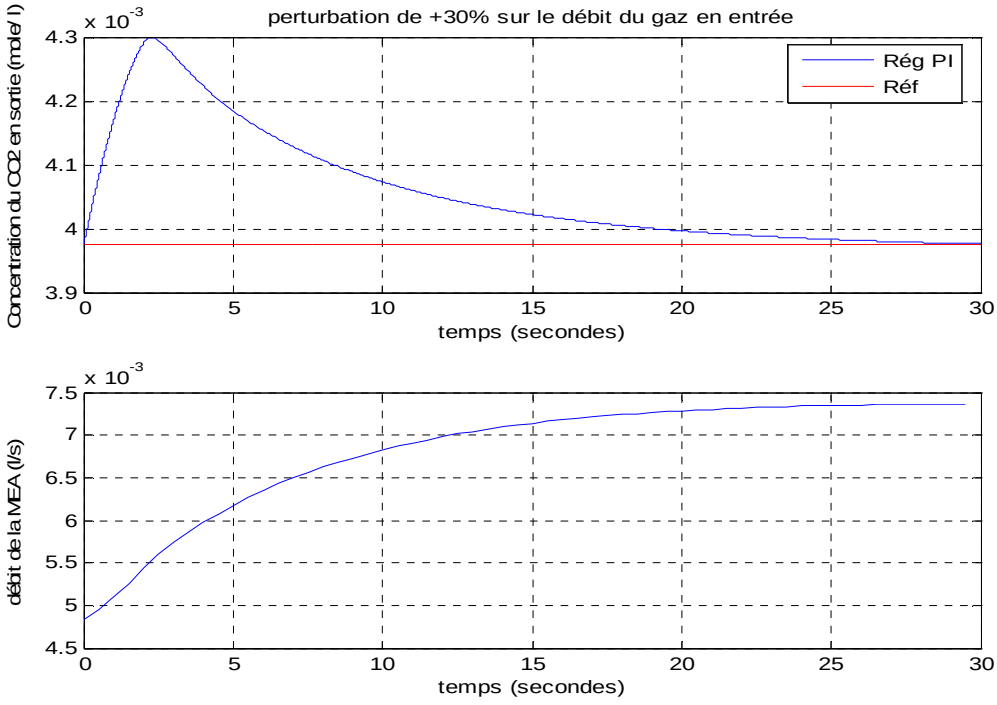


Figure 1.9. *Réponse pour une perturbation de +30% sur le débit du gaz en entrée.*

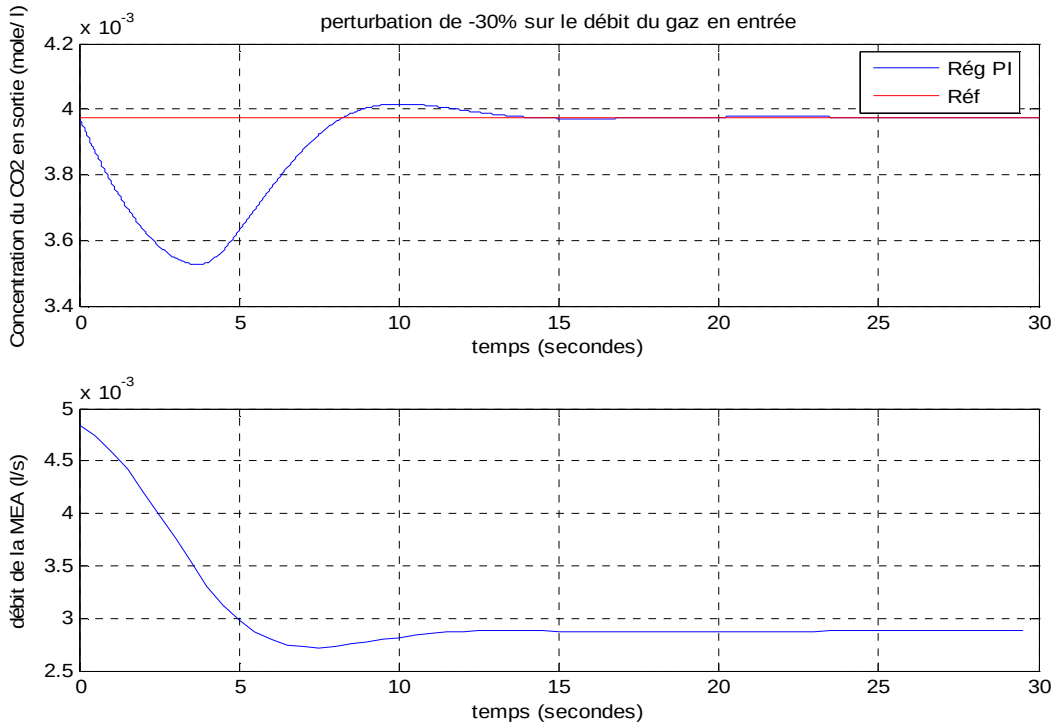


Figure 1.10. *Réponse pour une perturbation de -30% sur le débit du gaz en entrée.*

4. Discussion :

En boucle ouverte **:** on remarque d'après les différentes réponses que le système est stable en boucle ouverte avec une dynamique lente très ordinaire pour un processus chimique de diffusion. On remarque aussi la présence d'un temps mort en réponse à des perturbations en concentration qui est dû à la propagation de la différence de concentration tout au long de la colonne, ce qui n'est pas le cas pour les perturbations en débit, car une variation de débit à l'entrée de la colonne entraine presque instantanément sa variation en sortie.

En boucle fermée : le régulateur PI, rejette bien la perturbation et ramène la sortie à la référence mais lentement et tolère un dépassement considérable. Dans l'esprit d'améliorer les réponses à différentes perturbations on appliquera par la suite des lois de commande intelligentes basées sur la logique floue.

Chapitre II :

Commande Floue de la Colonne d'Absorption

I. Introduction à la logique floue (Fuzzy Logic) :

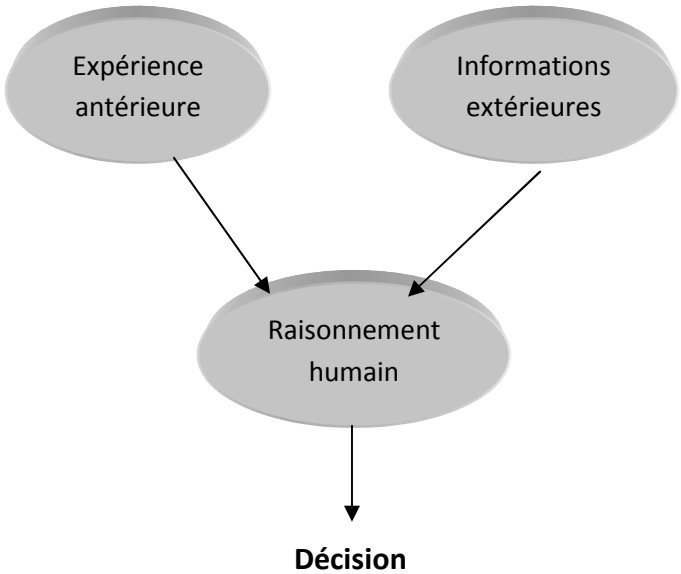
1. Introduction :

Depuis les années 1980, les systèmes basés sur la logique floue sont devenus l'un des domaines de recherche les plus fructueux en automatique. Les travaux contemporains se sont inspirés des recherches de ***Mamdani*** sur le contrôle flou, et ont été motivés par les articles de ***Lotfi ZADEH*** sur l'approche linguistique dans le réglage des systèmes. A partir de 1985 environ la logique floue est appliquée dans les systèmes de réglage ainsi que dans les systèmes experts, dans les systèmes de décision, pour la reconnaissance de formes …

La logique floue suscite actuellement un intérêt général de la part des chercheurs, des ingénieurs et des industriels, mais plus généralement de la part de tous ceux qui éprouvent le besoin de formaliser des méthodes empiriques, de généraliser des modes de raisonnement naturels, d'automatiser la prise de décision dans leur domaine, de construire des systèmes artificiels effectuant les taches habituellement prises en charge par les humains.

Le mode de pensée d'un être humain est généralement fondé sur un raisonnement empirique, où l'analogie et l'intuition jouent un rôle prépondérant. En fait, nos sens et notre jugement ne nous permettent d'évaluer certaines grandeurs que de manière imprécise ou vague : par exemple la température de l'eau de notre baignoire nous apparaîtra assez chaude ou très froide, ou comprise dans un certain intervalle de température, sans que nous puissions donner directement une valeur exacte de cette température.

Le schéma simplifié de ce type de raisonnement peut être représenté par :



Donc le but de la logique floue est d'approcher du raisonnement humain, et travailler non plus sur des conditions binaires mais sur des zônes. Elle sert donc à représenter des connaissances incertaines et imprécises.

La commande floue sert à prendre une décision même si l'on ne peut pas estimer les entrées/sorties qu'à partir de prédicats vagues ou lorsque ses entrées/sorties sont entachées d'erreurs que l'on peut évaluer que grossièrement.

On conçoit l'intérêt de faire entrer l'approche floue dans la régulation ou l'asservissement des processus industriels, pour lesquels les informations disponibles sont souvent imprécises, incertaines et parfois qualitatives, dans des boucles de régulation parfois incomplètes. Le savoir faire de l'opérateur, constitué entre autres souvent des règles simples, lui permet de conduire chaque machine plus correctement parfois qu'un algorithme classique.

2. Les bases de la commande par logique floue :

2.1 Définition :

La notion d'ensemble flou permet de définir une appartenance graduelle d'un élément a une classe, c'est-à-dire appartenir plus ou moins fortement à cette classe.

Un ensemble flou peut être vu comme une généralisation du concept du sous-ensemble ordinaire (sous-ensemble Booléen) dont la fonction d'appartenance prend seulement les deux valeurs 0 et 1 contrairement à un ensemble flou dont la fonction d'appartenance prend des valeurs dans l'intervalle $[0\ ,1]$.

2.2 Sous ensemble flou :

Les sous-ensembles flous (SEF) sont définis comme des ensembles pouvant contenir des éléments de façon partielle. Pour mieux comprendre ceci nous allons introduire un petit exemple de ***' Taille '*** illustré par la figure suivante :

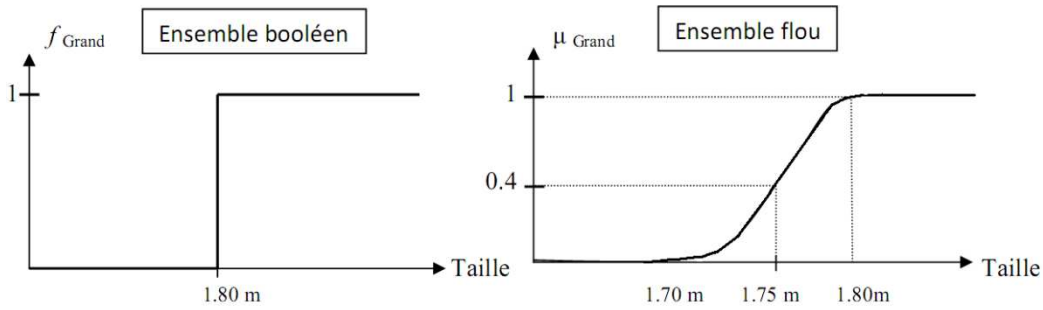


Figure 4.1 *Exemple de taille ensemble booléen et ensemble flou.*

On estime qu'un être est grand si sa taille égale ou dépasse 1,80m. Bien évidement une personne de 1,55m n'est pas grande alors qu'un individu de 1.95m l'est. Par contre, il est déraisonnable de considérer qu'un être de 1.81m est grand mais qu'un individu de 1.79m ne l'est pas du tout.

La notion de SEF constitue une sorte d'assouplissement de celle de sous ensemble classique, l'idée de base de ce concept est d'autoriser une ***graduation*** comprise entre 0 et 1 dans l'appartenance d'un élément à un sous ensemble. Un élément peut dès lors appartenir à un sous ensemble d'une manière non absolue, plus ou moins **[10] [11]**.

2.3 Caractéristiques d'un ensemble fou :

Soit un ensemble flou ***A*** d'un ensemble de référence ***X.***

- Le ***support supp(A)*** de ***A*** est constitué des éléments de ***X*** possédant au moins un peu le qualificatif flou défini par ***A*** :

$$SUPP\ (A) = \{x \in A; \mu_A(x) > 0\}$$

- Le ***noyau Ker(A)*** de l'ensemble flou ***A*** est le sous ensemble classique des éléments de ***X*** appartenant de façon absolue à ***A*** :

$$Ker\ (A) = \{x \in A; \mu_A(x) = 1\}$$

- ***La hauteur h(A)*** de ***A*** est le supremum de sa fonction d'appartenance μ_A :

$$h(A) = \sup_{x \in X} \mu_A(x)$$

- L'ensemble flou ***A*** est dit ***normalisé*** lorsque sa hauteur est égale à 1et son noyau diffère de l'ensemble vide.
- Nous appelons ***coup de niveau α*** ou ***α-coupe*** avec α= [0, 1] de l'ensemble ***A***, le sous ensemble classique A_α de ***X*** :

$$A_\alpha = \{x \in X ; \mu_A(x) \geq \alpha\}$$

Ces diverses caractéristiques sont illustrées par la figure suivante : **[12]**

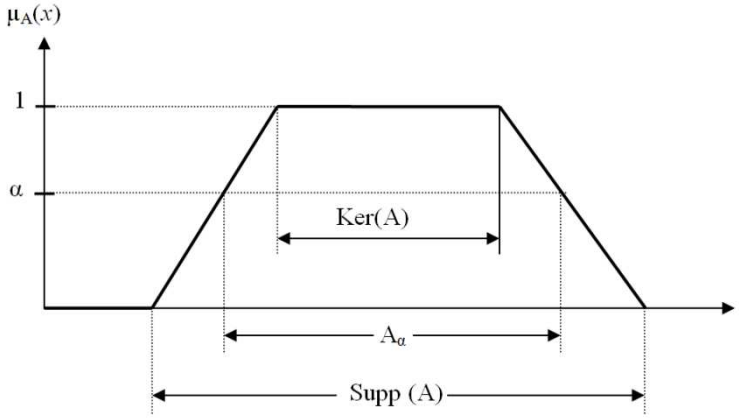


Figure 4.2 *Les différentes caractéristiques d'un ensemble flou.*

2.4 Fonctions d'appartenance :

On peut, pour toute variable floue ***x***, définir ce coefficient d'appartenance directement. Cette propriété se présente facilement par une ***fonction*** dite ***d'appartenance $\mu_A(x)$*** à valeurs dans [0,1], la notation signifie «coefficient d'appartenance de x à l'ensemble caractérisé par A», l'argument x se rattache à la variable linguistique et l'indice A désigne l'ensemble concerné.

2.5 Opérations sur les sous ensembles flous :

1- Deux ensembles flous A et B d'un même ensemble de référence X sont égaux, ce qui est écrit A=B, si leurs fonctions d'appartenance sont égales, autrement dit :

$$\forall x \in X ; \mu_A(x) = \mu_B(x)$$

2- Soient deux ensembles flous A et B de même ensemble de référence X, A est inclus dans B, et nous écrivons $A \subset B$ si pour tout *x* dans X : $\mu_A(x) \leq \mu_B(x)$

3- L'intersection de deux ensembles flous A et B d'un même ensemble de référence X est l'ensemble flou A ∩ B de X dont les éléments sont affectés du plus petit de leurs degrés d'appartenance : $\forall x \in X ; \mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\}$
C'est en faite la satisfaction simultanée de deux propriétés, on l'appel aussi l'opérateur « ***ET*** » **(Figure 4.3).**

4- L'union de deux ensembles flous A et B d'un même ensemble de référence X est l'ensemble flou $A \cup B$ de X dont les éléments sont du plus grand à leurs degrés d'appartenance : $\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\}$
Cet opérateur est appelé également « l'opérateur OU ». **(Figure 4.4)**

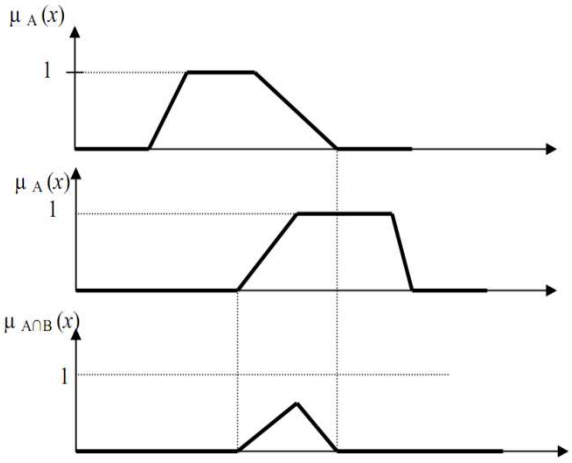


Figure 4.3 *: Opérateur ET (intersection).*

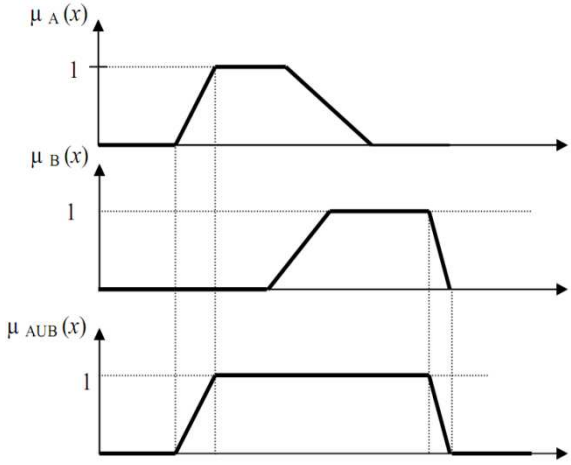


Figure 4.4: Opérateur OU (union).

5- Le complément d'un ensemble flou A de X est l'ensemble flou Ā de X dont la fonction d'appartenance est : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$
Il est appelé aussi l'opérateur « NON ». **(Figure 4.5)**.

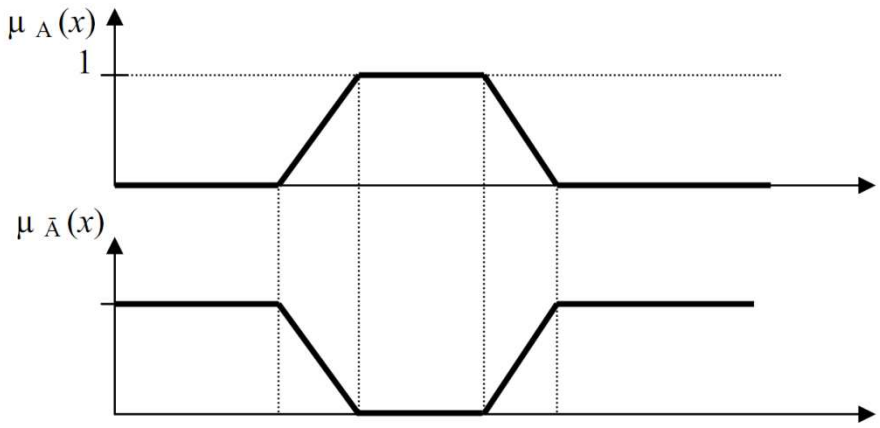


Figure 4.5 : *Opérateur NON (complément).*

Il existe également d'autres opérations mais sont moins utilisés :

- NON-OU (NOR) : A NOR B = 1 - max (A, B)
- OU EXCLUSIF (XOR) : A XOR B = (A OR B) AND NOT (A AND B) = A + B - 2 × min(A, B)
- NON-ET (NAND) : A NAND B = 1 - min (A, B)
- NON-XOR (NXR) : A NXR B = 1 + 2 × min (A, B) - (A + B)
- SUIVEUR (NOP) : NOP A = A. **[12]**

3. Réglage et commande par logique floue :

Comme déjà mentionné, un domaine d'application de la logique floue qui devient de plus en plus important, est celui du réglage et de la commande des processus industriels. En effet, cette méthode permet d'obtenir une loi de réglage souvent très efficace sans devoir faire des études théoriques approfondies **[13]**.

3.1-Analogie commande standard & commande floue :

a-Réglage avec régulateurs standard :

Le schéma synoptique suivant montre les étapes principales à suivre lors de la conception d'un régulateur classique (standard) :

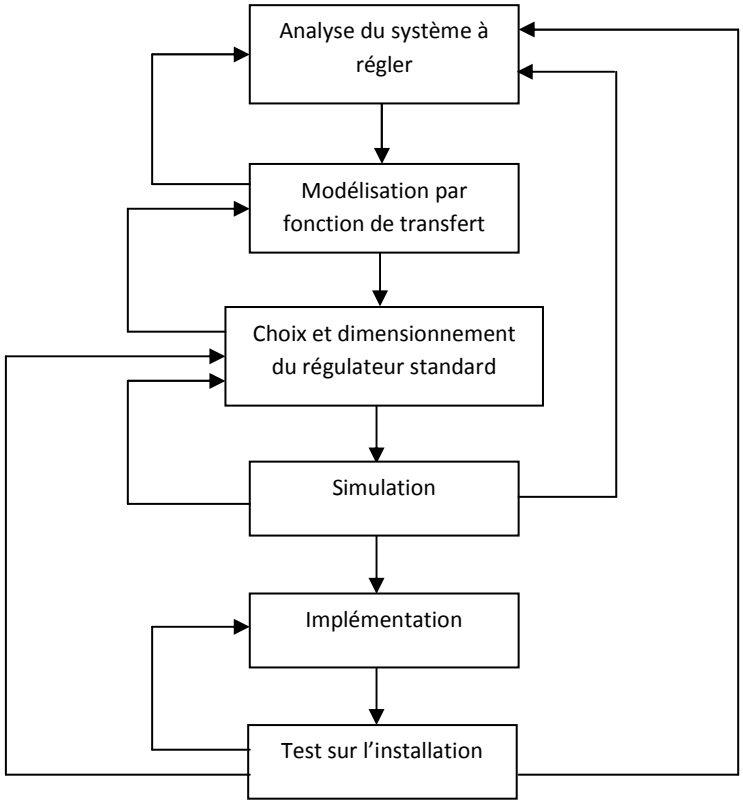


Schéma synoptique de la conception d'un régulateur standard.

b-Réglage avec logique floue :

Le procédé à suivre lors de la conception d'un réglage par logique floue est assez différent de celui d'un réglage standard, la figure suivante montre les étapes principales à suivre :

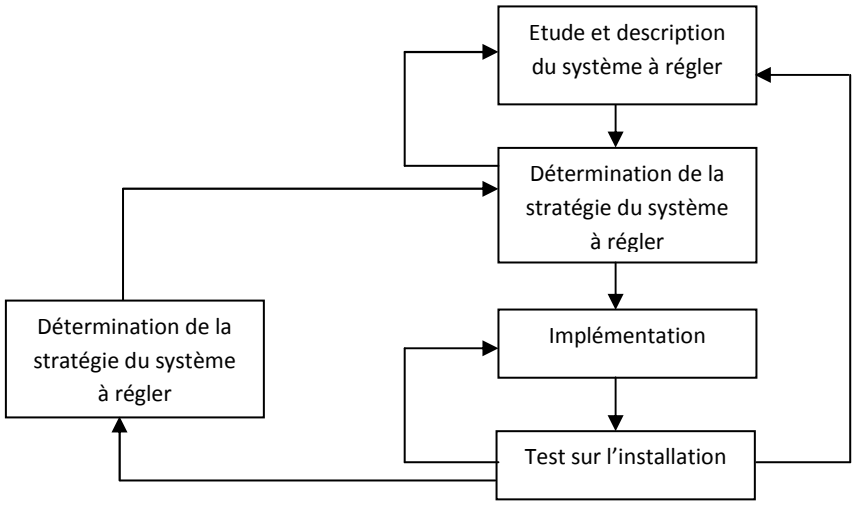


Schéma synoptique de la conception d'un régulateur flou.

c – Avantages et inconvénient du réglage par logique floue : [13]

Evidement, le réglage par logique floue réunit un certain nombre d'avantages et d'inconvénients. Les avantages essentiels sont :

- La non nécessité d'une modélisation.
- La possibilité d'implémenter des connaissances linguistiques de l'opérateur de processus.
- La maîtrise de systèmes à régler avec un comportement complexe (non linéaire ou difficile à modéliser).
- L'obtention fréquente de meilleures performances dynamiques.
- L'emploi possible aussi pour des processus rapides (grâce à des processeurs dédicacés).

Par contre, les inconvénients sont :

- l'impossibilité de la démonstration de la stabilité du circuit de réglage en toute généralité.
- la possibilité d'apparition de cycles limites à cause du fonctionnement non linéaire.

3.2-La commande floue :

Actuellement la commande floue est très répandue et trouve son application dans tous les domaines. Ceci s'explique fondamentalement par la difficulté de modélisation des systèmes réels de plus en plus sophistiqués et l'incapacité des commandes classiques à leurs assurer de bonnes performances. Un contrôleur flou en revanche n'exige pas de modèle du système et par sa qualité d'approximateur non linéaire universel, peut approcher le comportement d'un correcteur classique et même son extrapolation sur d'autres plages de fonctionnement. Le PI, PD, et PID flous en sont des exemples concrets **[14]**.

De nos jours la commande floue connaît un tel développement et diversification qu'on ne peut pas énumérer toutes les variantes existantes ; génétique-flou, neuro-flou, mode de glissement-flou,…etc.

- ***Principe d'une commande floue :***

La structure d'une commande floue, présentée par la **figure 4.6**, peut être décomposée en trois grands modules :

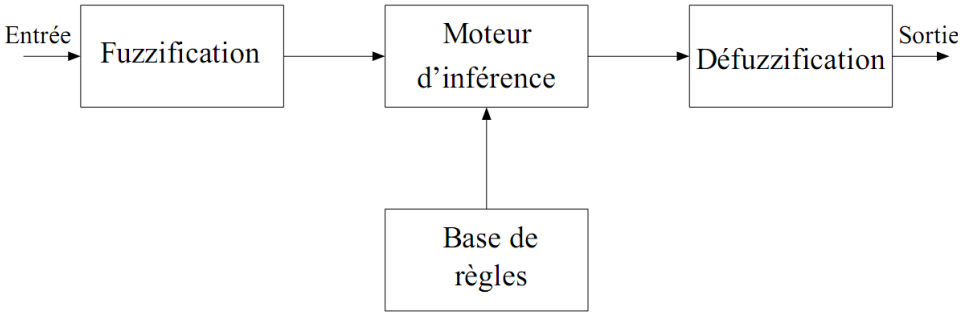


***Figure 4.6** : Structure générale d'une commande floue*

Le premier de ces modules traite les entrées du système : c'est la fuzzification. Il permet d'associer à chacune des entrées réelles, par le biais de fonctions d'appartenances, un degré d'appartenance pour chacun des sous-ensembles flous définis sur l'ensemble du discours.

Le deuxième module est constitué du moteur d'inférence et de la base de règles. Celle-ci est constituée de règles de type : ***"Si..., Alors..."*** et va permettre de passer des degrés d'appartenance des grandeurs d'entrées aux degrés d'appartenance aux sous-ensembles flous de la grandeur de commande. Le moteur d'inférence, lui, va permettre de générer une conclusion à partir des entrées et des règles actives. Il calcule alors les degrés d'appartenance aux sous-ensembles flous correspondant à la commande du système.

Enfin, le dernier module, l'interface de défuzzification, va permettre de transformer les degrés d'appartenance des sous-ensembles flous de commande en grandeur numérique. C'est la transformation inverse du module de fuzzification **[16]**.

Il existe plusieurs méthodes de défuzzification.

- **Méthode de Mamdani :**

Mamdani fut le premier à utiliser la logique floue pour la synthèse de commande, il utilise le minimum comme opérateur de conjonction et d'implication. Les règles correspondent au type :

Si x_1 est A_1^i et …… x_n est A_n^i Alors y est B^i

Où B^i est un SEF (donc les conclusions sont des SEF)

En général, les B^i forment une partition de l'espace de sortie. L'inférence floue correspond aux étapes suivants pour un vecteur d'entrée x=(x_1 …… x_n) :

1- Calcul du degré d'appartenance de chaque entrée aux différents SEF : $\mu_{A_j}^i(x_j)$ j=1 à n, i=1 à N.

2- Calcul de la valeur de vérité de chaque règle pour i=1 à N :

$\alpha_i(x) = \min_j(\mu_{A_j}^i(x_j))$ j=1 à n

3- calcul de la contribution de chaque règle : $\mu_i(y) = \min(\alpha_i(x), \mu_B{}^i(y))$

4- Agrégation des règles : $\mu(y) = \max_i(\mu_i(y))$

Le résultat est donc un sous ensemble flou caractérisé par sa fonction d'appartenance. Pour obtenir une conclusion « nette » il faut défuzzifier, la méthode du centre de gravité donne alors :

$$y = \frac{\int u.\mu(u)\,du}{\int \mu(u)\,du}$$ Ou en échantillonné : $$y = \frac{\sum_k u_k.\mu(u_k)}{\sum_k \mu(u_k)}$$

Cette méthode est parfois appelée « min-max-barycentre », il existe plusieurs variantes de la méthode de Mamdani, on trouve par exemple : « produit-somme-barycentre », et toute ces variantes donnent des SEF en sortie, autrement dit la méthode de Mamdani utilise des SEF pour les conclusions, d'où la nécessité de l'étape de défuzzification.

- **Méthode de Sugeno (Takagi-Sugeno : TS):**

La méthode de Sugeno constitue un cas particulier très important. Elle a apparu en 1975 et elle est beaucoup répandue dans la théorie de contrôle des procédés. A la différence des règles floues standards, la conclusion n'est pas représentée par un SEF mais par une valeur constante (singleton) qui est une fonction linéaire des entrées.

Si x_1 est A_{1i} et ……. x_n est A_{ni} Alors $y = C_0 + C_1*x_1+……+C_n*x_n$

Où les C_i sont des constantes constituant la table des règles.

3.3 Synthèse d'un FLC :

Un contrôleur flou FLC (Fuzzy Logic Controller) est un système flou avec les modules expliqués précédemment ***(Figure 4.7)***

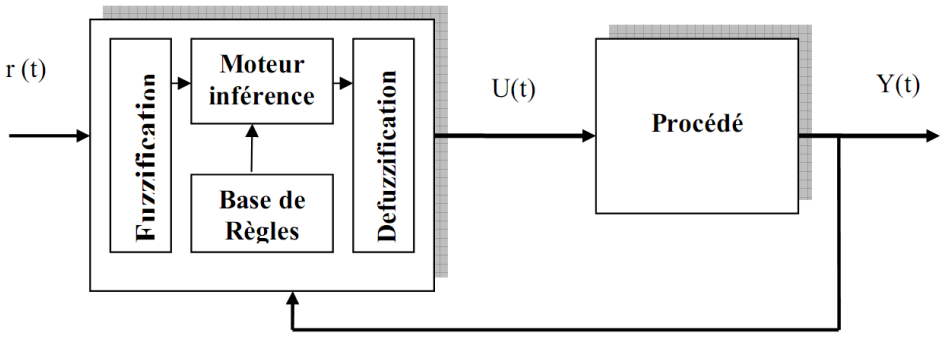


***Figure 4.7** : Structure d'un FLC*

La synthèse d'un FLC passe par les étapes suivantes :

- **Obtention de la base de connaissance :** c'est la partie la plus importante de la synthèse.
- **Le choix des entrées et sorties :** un choix d'entrées très particulièrement utilisé est l'erreur et sa variation avec si besoin des traitements comme le filtrage, l'intégration…etc.
- **Choix des ensembles flous :** leur nature et leur nombre peut varier suivant la plage de variation des entrées et sorties, la dynamique du système à commander…
- **Choix des opérateurs flous :** selon le but recherché et les contraintes évoquées.
- **Les gains de normalisation :** il est préférable de travailler dans un domaine normalisé [0,1].

II. Commande floue appliquée à la colonne d'absorption:

Pour améliorer les performances par rapport à la régulation PI, une commande floue sera proposée. En effet, la commande par logique floue semble une alternative très intéressante vue sa robustesse, sa simplicité et le grand succès qu'elle a réalisé en pratique. La problématique fondamentale dans la commande par logique floue est la synthèse du contrôleur, pour ceci, dans un premier temps un régulateur flou « ***standard »*** sera proposé.

Le régulateur flou choisi ici, est un régulateur de type ***Takagi-Sugeno*** donc la table des conclusions sera formée par des constantes. Dans ce qui suit, on explicitera le schéma général de la commande, le nombre et la formes des fonctions d'appartenance, la dimension de la table de décision,…

1. Schéma général de la commande :

La plupart des contrôleurs flous rencontrés dans la littérature prennent comme entrées, l'erreur e et sa variation de . Le schéma suivant montre les différents éléments constituant le contrôleur ainsi que la boucle fermée de régulation.

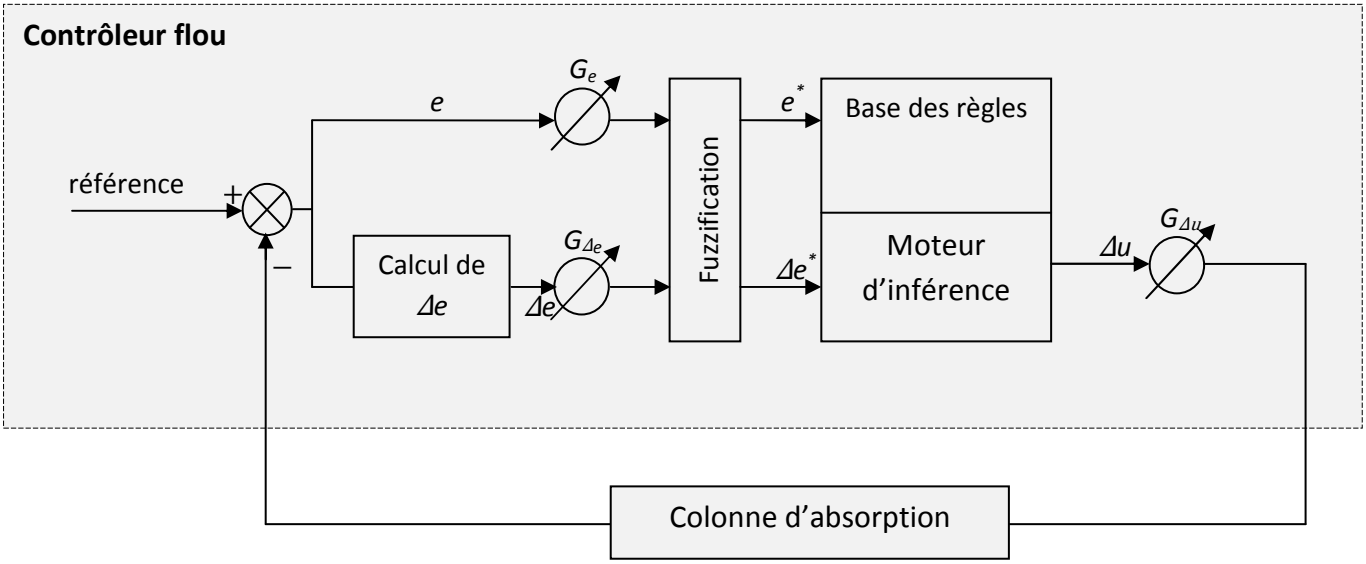


Figure 4.8. *Schéma d'un FLC en boucle fermée.*

Bloc calcul de de : pour calculer la variation de l'erreur, on adopte l'approximation

$$de = e(k) - e(k-1)$$

C'est-à-dire, on admet que la variation de l'erreur est égale à l'erreur à l'instant k moins l'erreur à l'instant $k-1$.

Les gains de normalisation : les gains $G_e \; et \; G_{de}$ ont pour rôle de normaliser les deux variables $e \; et \; de$ sur l'univers de discours normalisé $[-1, 1]$. Le gain $G_{\Delta u}$ permet la mise en échelle du signal de commande.

Un bloc de fuzzification : génère les variables floues correspondantes aux deux variables numériques $e \; et \; de$.

Table de décision et moteur d'inférence : réalisant les opérations logiques sur les variables floues.

Remarque : le bloc de défuzzification n'apparait pas car la structure est celle de ***Takagi-Sugeno***.

2. Fonctions d'appartenances :

Pour notre problème nous avons choisie des fonctions d'appartenances gaussiennes pour les deux entrées erreur et variations de l'erreur. En effet les fonctions gaussiennes sont continues et peuvent être codées que par deux paramètres, le centre et l'écart-type. Après plusieurs essais, il s'est avéré que le choix de sept fonctions d'appartenances pour les deux entrées semble le bon.

Les fonctions d'appartenances sont prises symétriques, uniformément réparties sur l'univers de discours de manière à le recouvrir en entier avec un chevauchement de 50%, elles sont symbolisées par :

PG : Positif Grand, ***PM*** : Positif Moyen, ***PP*** : Positif Petit, ***Z*** : Zéro, ***NP*** : Négatif Petit, ***NM*** : Négatif Moyen et ***NG*** : Négatif Grand.

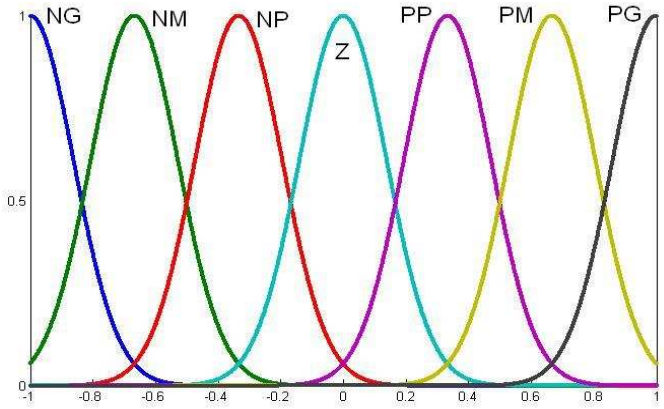


Figure 4.9. *Fonctions d'appartenances pour l'erreur e et la variation de l'erreur de .*

3. Table de décision :

La table de décision adoptée pour notre contrôleur est celle universelle de ***Mac Vicar-Whelan***, illustré par la table **4.1.** Elle est organisée sous la forme d'une table diagonale symétrique de ***49*** règles décisionnelles composées par les paires situation/action de la forme :

$$IF \quad e \; is \; Z \quad and \quad de \; is \; Z \quad THEN \quad \Delta u = 0$$

Δu		e						
		NG	NM	NP	Z	PP	PM	PG
Δe	NG	-1	-1	$-2/3$	$-2/3$	$-1/3$	$-1/3$	0
	NM	-1	$-2/3$	$-2/3$	$-1/3$	$-1/3$	0	$1/3$
	NP	$-2/3$	$-2/3$	$-1/3$	$-1/3$	0	$1/3$	$1/3$
	Z	$-2/3$	$-1/3$	$-1/3$	0	$1/3$	$1/3$	$2/3$
	PP	$-1/3$	$-1/3$	0	$1/3$	$1/3$	$2/3$	$2/3$
	PM	$-1/3$	0	$1/3$	$1/3$	$2/3$	$2/3$	1
	PG	0	$1/3$	$1/3$	$2/3$	$2/3$	1	1

Table 4.1. *Table de décision de* ***Mac Vicar-Whelan****.*

4. Algorithme de commande :

Le calcul de la commande se résume par les étapes suivantes :

1- Calcul de l'erreur e et de sa variation Δe .

$$e(kT_e) = sortie - référence$$

$$\Delta e(kT_e) = \frac{e(kT_e) - e((k-1)T_e)}{T_e}$$

2- Normalisation de l'erreur et de sa variation à l'aide des gains correspondant :

$$e^*(kT_e) = G_e \times e(kT_e)$$

$$\Delta e^*(kT_e) = G_{\Delta e} \times \Delta e(kT_e)$$

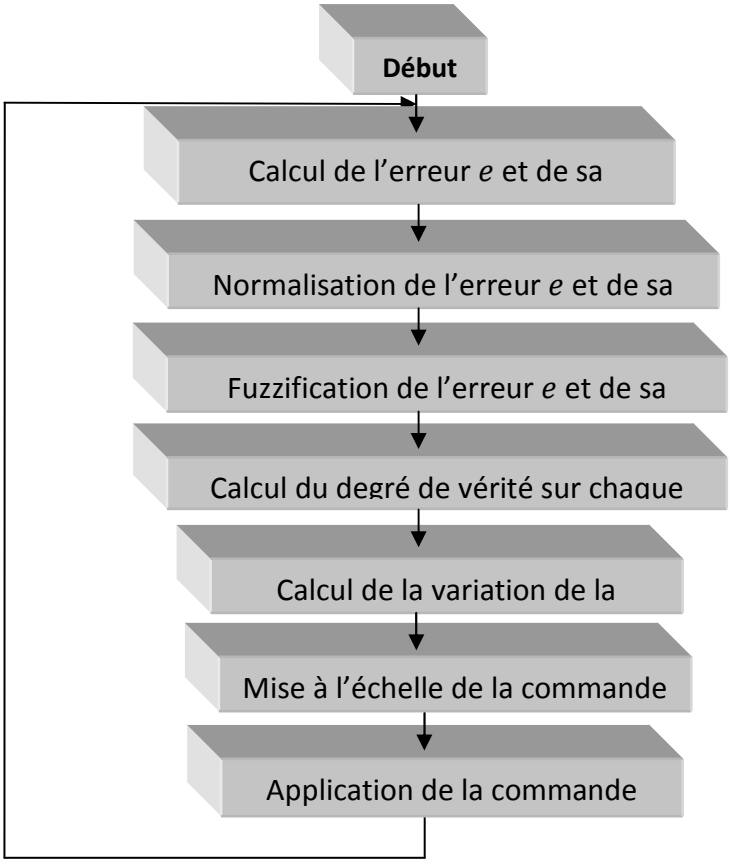
3- Fuzzification qui consiste en la conversion des variables normalisée en variables floues.

4- Calcul du degré de vérité sur chaque sous ensemble flou, en appliquant la fonction d'appartenance correspondante.

5- Calcul de la variation de la commande Δu par projection des vecteurs degrés de vérité sur la base des règles.

6- Mise à l'échelle de la commande à l'aide du gain $G_{\Delta u}$.

Les étapes précédentes sont montrées sur l'organigramme suivant :



Algorithme de commande par logique floue.

5. Simulation en boucle fermée de la commande floue :

Dans ce paragraphe, nous allons présenter les résultats de simulation du régulateur flou standard en boucle fermée, décrit précédemment, par rapport aux différentes perturbations sur la concentration du CO_2 et sur le débit du gaz en entrée, puis nous allons donner une comparaison entre cette technique de commande floue avec la régulation classique PI donnée dans le premier chapitre.

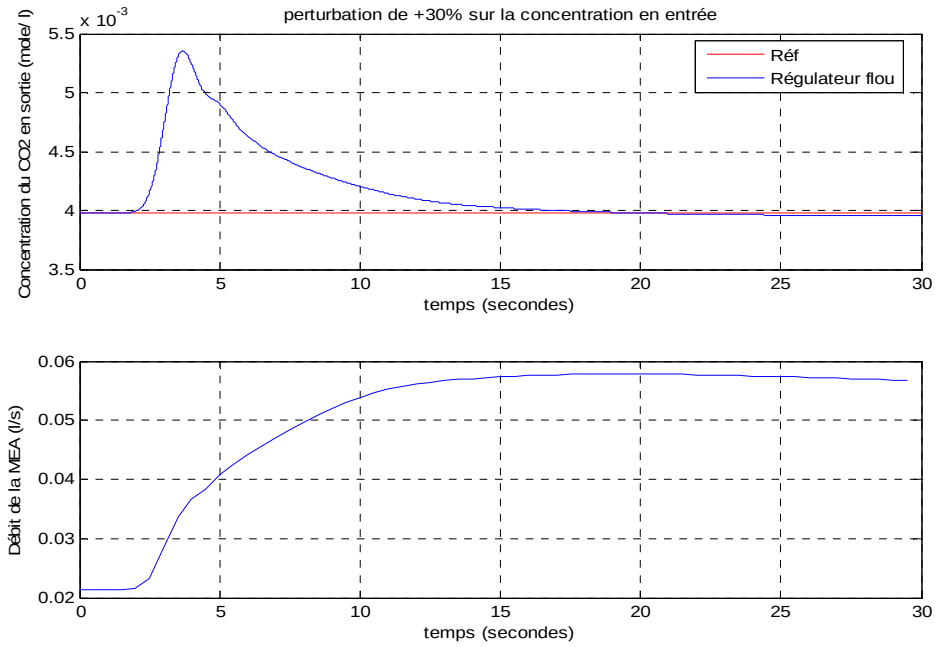


***Figure 4.10.**: Réponse pour une perturbation de +30% sur la concentration du gaz en entrée.*

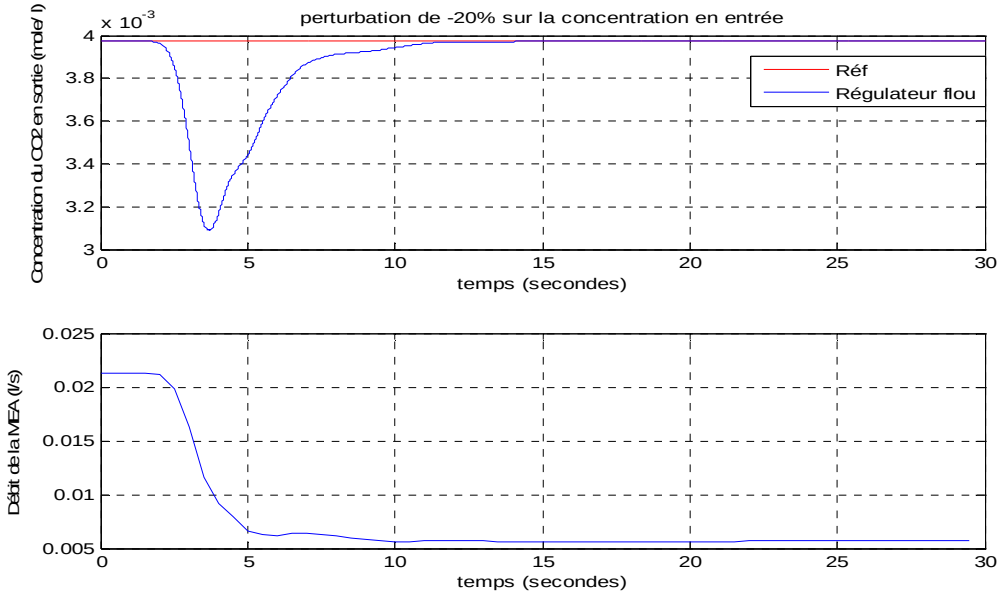


***Figure 4.11.** Réponse pour une perturbation de -20% sur la concentration du gaz en entrée.*

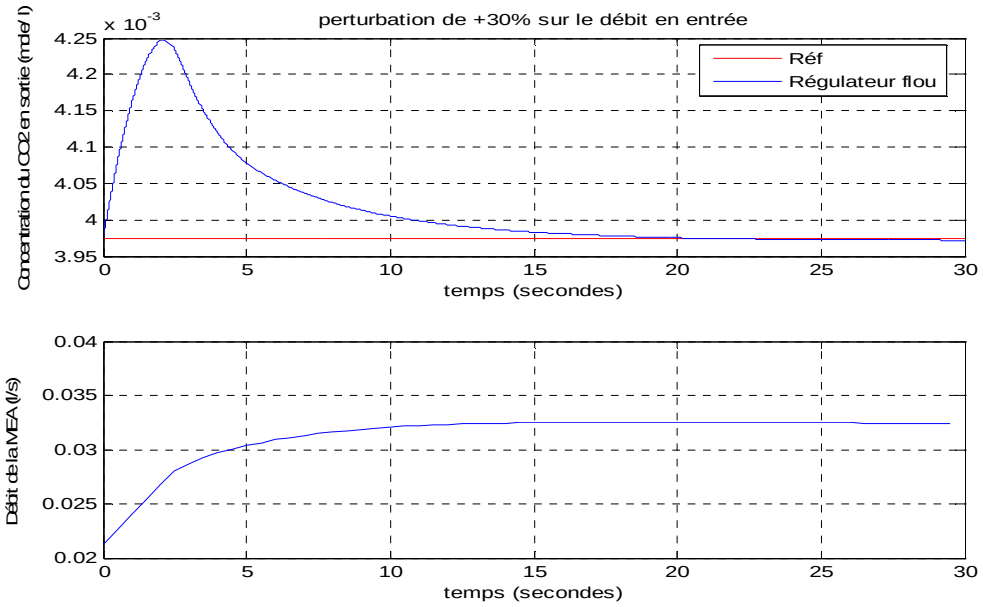


Figure 4.12. *Réponse pour une perturbation de +30% sur le débit du gaz en entrée.*

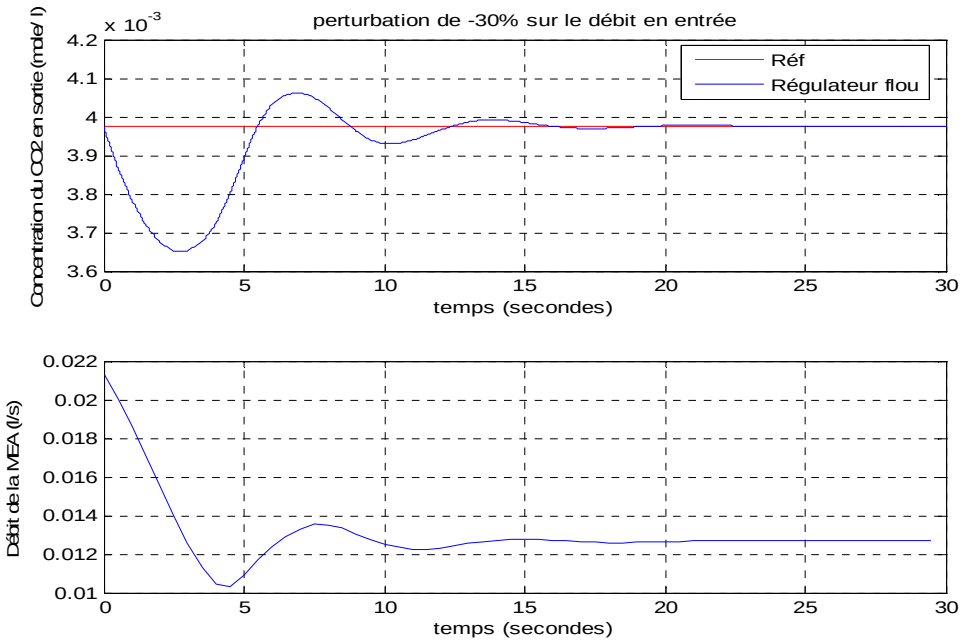


Figure 4.13. *Réponse pour une perturbation de -30% sur le débit du gaz en entrée.*

- **<u>Comparaison du régulateur flou avec le PI :</u>**

Ici on donne une comparaison floue – PI, la ***figure 4.14*** suivante illustre bien ceci :

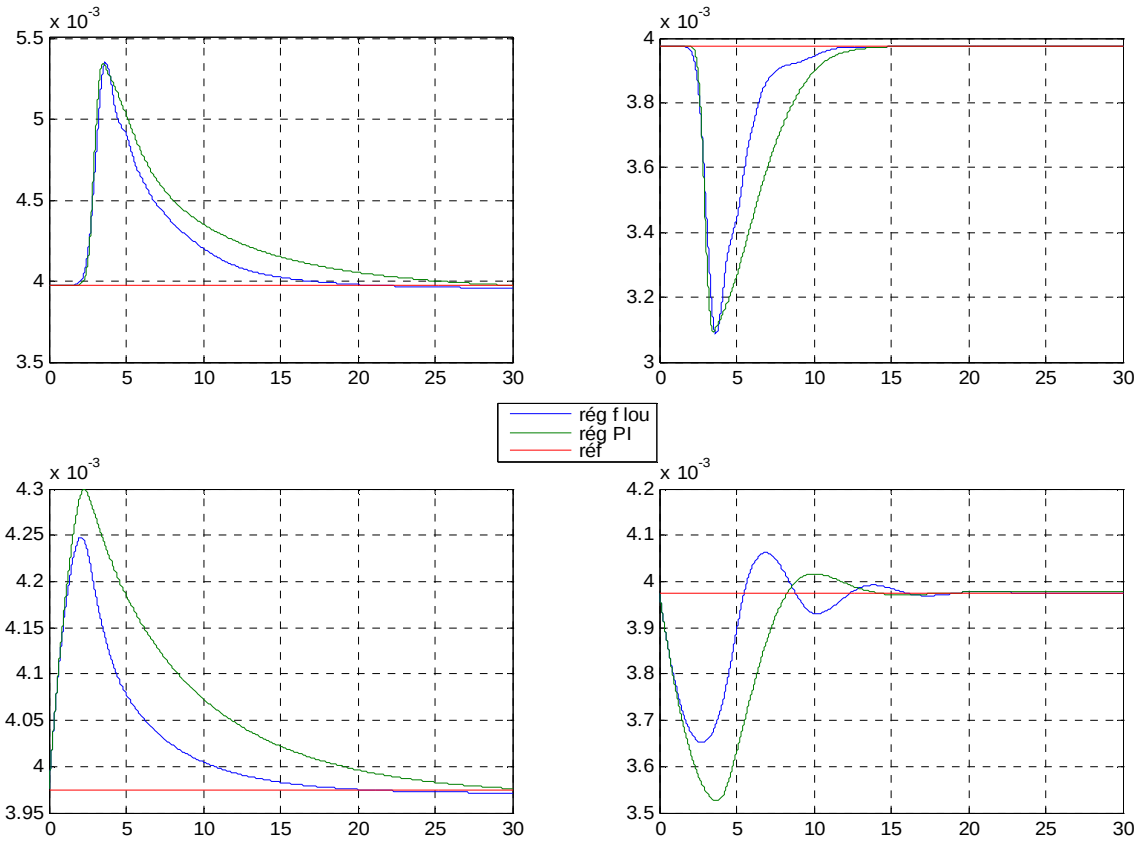


Figure 4.14. *Réponses pour les différentes perturbations, commande floue et PI.*

On remarque qu'avec le régulateur flou, les performances sont améliorées. Le temps de réponse est nettement moins important. Le premier dépassement a été réduit pour les perturbations en débit, par contre pour les perturbations en concentrations il est resté tel quel, néanmoins l'erreur en réponse a été réduite pour toutes les perturbations.

L'avantage de la commande floue ne se résume pas à l'amélioration des performances mais elle a d'autres avantages tels que la robustesse, la simplicité de l'implémentation et de la mise en œuvre pratique.

- **Conclusion :**

Comme cela a été déjà relevé dans les paragraphes précédents, une commande floue est en quelque sorte une modélisation linguistique du régulateur traduisant des connaissances d'opérateurs et d'experts.

La philosophie d'une commande floue est donc très différente de l'approche classique, qui nécessite, pour la synthèse et l'analyse d'un régulateur, un modèle mathématique du système à commander.

Un régulateur flou mérite en particulier d'être considéré quand de riches connaissances linguistiques sur la manière de piloter un processus difficile à modéliser sont à disposition.

Les correcteurs flous de type Takagi-Sugeno possèdent de nombreux paramètres et la problématique est le réglage de ceux-ci, malheureusement il n'existe pas une méthode systématique qui traite cela. Pour remédier à ce problème, il est nécessaire de choisir une méthode d'optimisation multidimensionnelle qui doit être efficace de telle façon à avoir une structure optimale pour notre contrôleur flou, et cela sera l'objet des chapitres qui suivent.

Chapitre III :

Les Algorithmes Génétiques

I. Algorithmes génétiques :

1. Introduction :

Le problème de l'optimisation :

L'optimisation itérative est aussi veille que la vie. Même des êtres très primitifs agissent selon une pulsion simple qui peut être résumée en quelques mots : « améliorer sa situation ». A partir de là, bien sur, de nombreuses stratégie sont imaginables, mais celles que nous voyons tous les jours en action dans la nature, et qui font la preuve de leurs efficacité par la persistance des espèces qui les pratiquent, nous offrent un large éventail de solutions. Il n'est donc pas étonnant que, explicitement ou implicitement, plusieurs modèles mathématiques d'optimisation s'inspirent de comportement biologique et usent abondamment de métaphore et de termes issus de la génétique, de l'ethnologie voir de l'ethnologie ou de la psychologie.

En matière d'optimisation, certains problèmes sont plus difficiles que d'autres. C'est le cas, entre autres, des problèmes combinatoires. Très souvent la notion de difficulté est plus ou moins liée à l'état de l'art en matière d'algorithme de recherche : on ne sait pas résoudre tel problème ou il faut un temps considérable, donc il est difficile **[36]**.

Méthodes d'optimisation heuristiques :

- **Recuit simulé :** issu du mécanisme du recuit appliqué en métallurgie.
- **Recherche tabou :** c'est un recuit avec un mécanisme de mémorisation ressemblant à le mémoire humaine.
- **Colonies de fourmis :** l'étude du comportement des colonies de fourmis à l'origine de l'idée de l'algorithme.
- **Marche aléatoire :** algorithme stochastique aveugle.
- **Essaims particulaires :** transduction de la manière dont les essaims d'abeilles coopèrent.
- **Algorithmes génétiques :** largement utilisés, inspirés de l'évolution génétique des organismes vivants, qui seront l'objet de ce chapitre. **[33]**.

2. Les algorithmes génétiques :

2.1 Métaphore :

Les algorithmes génétiques sont des algorithmes heuristiques faisant partie de la famille des algorithmes évolutifs inspirés directement des travaux de Charles Darwin sur l'évolution des espèces. Dans son célèbre livre :"*origins of species*" les fondements de la théorie de l'évolution naissent, le principe étant le suivant : Dans un environnement donné les individus les plus adaptés survivront, alors les espèces évoluent dans le sens d'être de plus en plus adaptées à leur environnement.

Les algorithmes génétiques ont déjà une histoire relativement ancienne puisque les premiers travaux de John Holland sur les systèmes adaptatifs remontent déjà à 1962, l'ouvrage de ***David Golberg*** a largement contribué à leur vulgarisation.

Ils connaissent depuis les années 1990 un développement considérable, notamment suite à l'apparition des architectures massivement parallèles exploitant leur parallélisme intrinsèque. Aujourd'hui sont d'un intérêt d plus en plus croissant vu les résultats pratiques qu'ils ont donnés en plus de quelques résultats théoriques qui donnent un fondement mathématique à ces techniques d'optimisation.

2.2 Structure de base :

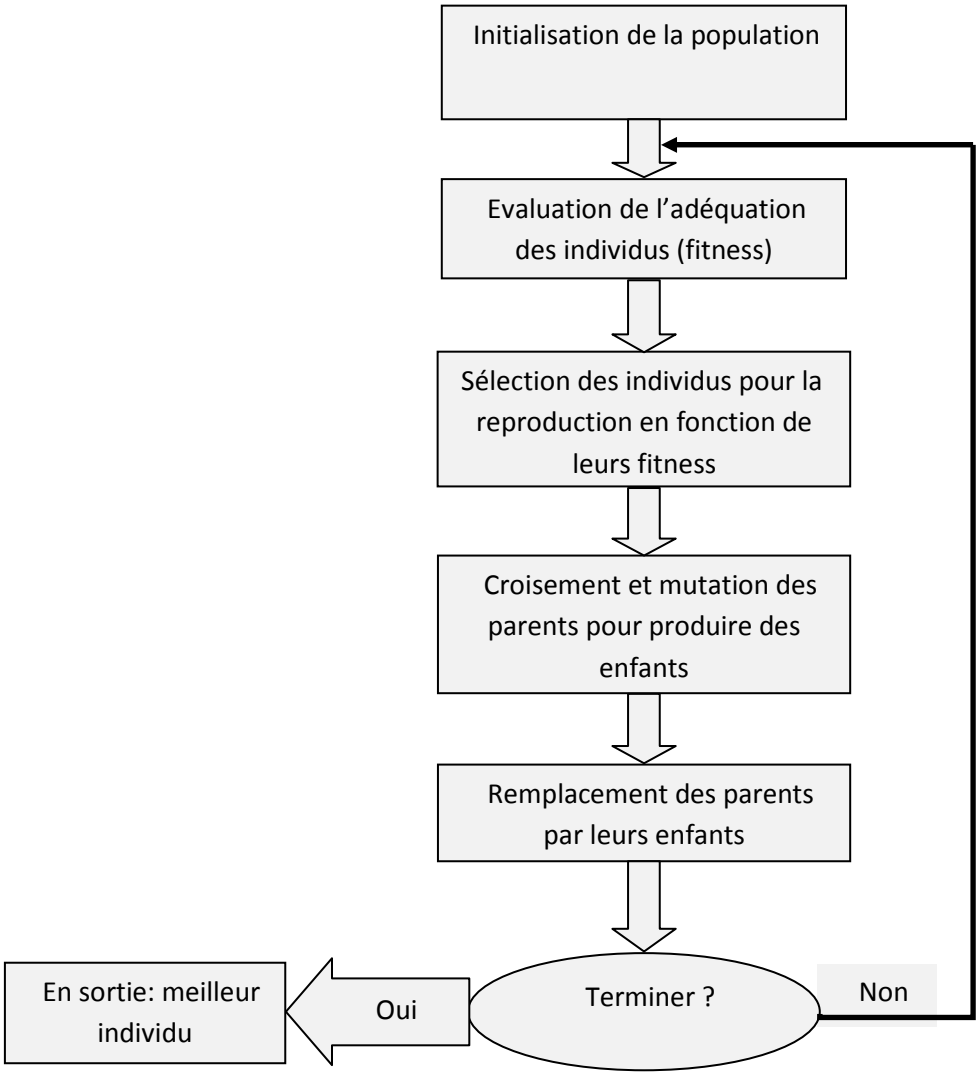


Figure 3.1. *Structure de base d'un algorithme génétique*.

2.3 Définition des éléments constituant un algorithme génétique :

2.3.1 Codage :

Un principe de codage des paramètres du problème posé. Cette opération associe à chacun des points de l'espace de recherche une structure donnée qui est appelé individu. Elle se place généralement après une phase de modélisation mathématique du problème traité. La qualité du codage des données conditionne fortement le succès de l'algorithme. Les codages binaires ont été très utilisés à l'origine mais les codages réels sont désormais largement utilisés, notamment pour l'optimisation de problèmes à variables réelles.

2.3.2 Population :

La population est l'ensemble d'individus (ou chromosomes) qu'on doit faire évoluer à chaque génération en appliquant les différentes opérations décrites dans la structure générale.

Un mécanisme de génération de la population initiale :

Ce mécanisme doit être capable de générer une population non homogène uniformément repartie sur l'espace de recherche dans le cas où on aucune idée de l'optimum, sinon, dans le cas où on sait dans quelle région de l'espace de recherche l'optimum sera trouvé, on génère la population dans cette région réduite. Un choix pareil accélérera considérablement la convergence de l'algorithme.

2.3.3 Une fonction à optimiser :

Celle-ci retourne une valeur dans $\mathbb{R}^+$ appelée fitness ou fonction d'évaluation de l'individu. En générale on ramène les problèmes à une maximisation pour rester conforme avec le principe de la sélection naturelle.

2.3.4 Opérateurs de reproduction :

Simulant la reproduction biologique qui est directement inspirée de deux opérateurs génétiques biologiques : croisement et mutation. Le croisement permet la production de nouveaux éléments en croisant d'autres déjà existant ainsi il assure l'héritage du code génétique des ancêtres, l'autre c'est la mutation qui a généralement un effet négatif sur l'individu biologique, pour l'algorithme la mutation garantit l'ergodicité de la recherche "aucune direction de recherche n'est vraiment privilégiée" ainsi une bonne exploration de l'espace de recherche sera garantie.

2.3.5 Critère d'arrêt :

Il n'y a pas un critère d'arrêt garantissant la convergence mais le critère le plus largement utilisé est la limitation du temps d'exécution ou du nombre maximum de générations. On peut utiliser un critère d'arrêt comme la stationnarité de la population c'est à dire : si la population n'évolue pas au bout d'un certain nombre de génération fixé on arrête l'exécution, en faite c'est une bonne raison d'arrêter l'exécution car soit l'algorithme a

convergé ou qu'il ne peut plus améliorer la population. Dans ce derniers cas, il se peut que l'algorithme soit tombé dans un optimum local et il ne puisse plus en sortir.

2.3.6 Paramètres de dimensions :

La taille de la population est soit constante soit variable au fil des générations, en générale elle est prise constante. Les probabilités de croisement et de mutation ainsi que le nombre maximum de génération et la longueur des chromosomes jouent un rôle dans la convergence de l'algorithme.

2.4 Description détaillée :

2.4.1 Codage :

Historiquement les premiers Algorithmes génétiques utilisaient des chromosomes codés sous formes de chaînes de bits, ce type de codage permet la définition d'opérateurs de croisement et de mutation simples. C'est également avec ce type de codage que les premiers résultats théoriques de convergences ont été obtenus. Néanmoins ce type de codage n'est pas toujours bon car :

- Deux éléments voisins en termes de distance de ***Hamming*** ne codent pas obligatoirement deux éléments proches de l'espace de recherche, cet inconvénient peut être évité en utilisant un codage Gray.
- Pour des problèmes de grande dimension chaque variable est représentée en générale par une chaîne binaire faisant partie du chromosome, alors cette représentation ne reflète pas la vrai structure du problème. L'ordre des variables ayant une importance dans la structure du chromosome alors qu'il ne l'est pas forcément dans la réalité du problème.
- Pour une plus grande précision numérique le chromosome sera plus long et un temps de calcul plus important sera nécessaire.
- ✓ Le codage réel évite ce genre de problème mais il pose le problème de trouver des opérateurs génétiques (croisement et mutation) plus adaptés, chose qui n'est pas très facile vu que les performances de l'algorithme sont intrinsèquement liées à ces opérateurs.

2.4.2 Gestion des contraintes et génération de la population initiale :

Un élément de la population qui viole une contrainte se verra attribuer une mauvaise fitness et aura une probabilité forte d'être éliminé par le processus de sélection. Cependant quand l'optimum peut être sur la frontière de l'espace de recherche il sera intéressant de conserver les éléments non admissibles car ils peuvent permettre de générer des éléments admissibles de bonnes qualités. Toute fois il est un peu difficile de gérer ces contrainte tout en ne favorisant la recherche de solutions admissible au détriment de la recherche de l'optimum ou inversement. Disposant d'une population d'individus non homogènes, la diversité doit être entretenue au cours des générations à fin de parcourir le plus largement possible l'espace de recherche, c'est le rôle de la sélection ainsi que les opérateurs génétiques.

2.4.3 Fonction fitness :

À l'inverse d'algorithmes d'optimisation dits classiques les Algorithmes génétiques ne requièrent pas d'hypothèses particulières sur la régularité de la fonction à optimiser notamment pas ses dérivées successives. La seule condition est que la fonction objectif "fitness" soit calculable en n'importe quel point de l'espace de recherche, alors leurs domaines d'application sont très vastes. Le peu d'hypothèses requises permet de traiter des problèmes très complexes, la fonction objectif peut être ainsi le résultat de la simulation, on peut citer quelques exemples rencontrés en pratique :

« On dit d'un problème qu'il est discret ou combinatoire si l'espace de recherche est discret, sinon si l'espace de recherche est continu on dit qu'il est continu, à noter qu'on peut ramener quelques problèmes continus en problèmes combinatoires à savoir le codage binaire d'un problème à variables réelles par exemple. »

- Recherche des paramètres des régulateurs (flou par exemple) où la fonction fitness est le résultat d'une simulation.
- Problèmes d'identification des processus.
- Gestion des tâches ou ordonnancement sur une machine parallèle.
- Recherche d'un cycle *Hamiltonien* optimal sur un graphe : problème classique du voyageur de commerce.
- Placement de composant électroniques et architectures des processeurs….etc.

D'après ces quelques exemples on voit la diversité des problèmes traités par algorithmes génétiques. On note que les algorithmes génétiques sont de nature combinatoire mais leur extension aux problèmes continus a donné de bons résultats, des fois on préfère ramener un problème continu en un problème combinatoire comme c'est le cas du codage binaire d'un problème à variables réelles **[25, 26]**.

2.4.4 Sélection :

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre plus ou moins important de principes de sélection, on peut citer :

- La roue de la fortune (roulette biaisée, Roulette Wheel Selection).
- Stochastic remainder without replacement selection.
- Sélection par tournoi.

Ces quelques techniques de base peuvent être modifiées pour répondre mieux à un problème considéré.

- **Problème de la sélection :**

La fonction de sélection doit choisir les bons individus tout en ne pas négligeant totalement les mauvais, vu que, le croisement d'un bon individu avec un mauvais peut donner un très bon individu, autrement dit, un individu même s'il est mauvais peut avoir quelques bon gènes. Les deux problèmes qui peuvent se poser sont les suivant :

➢ Problème de convergence prématurée : si la fonction de sélection favorise beaucoup plus les bons individus alors au bout de quelques générations la population sera uniforme et elle n'évoluera plus.

➢ Convergence très lente : si la fonction de sélection ne favorise que très peu les bons individus alors le total de la population ne sera amélioré que très peu ou pas du tout. L'algorithme peut ne pas converger !

- **Sélection par tournoi :**

Dans cette technique on organise une sorte de tournois entre les individus deux à deux ou plus et puis le gagnant sera choisi avec une probabilité bien déterminée. On peut appliquer des méthodes plus performantes de la théorie des jeux comme la stratégie de Nash.

- **Sélection par la roulette biaisée :**

C'est la technique de base la plus largement exploitée. Elle consiste à associer à chaque individu un secteur de longueur proportionnel à son fitness et puis on tourne la roue autant de fois que le nombre d'individus qu'on veut sélectionner, à chaque tirage l'élément choisis et celui désigné par le curseur (figure). Il est évidant qu'un bon individu, ayant donc un secteur plus long, a plus de chance d'être choisi **[24]**.

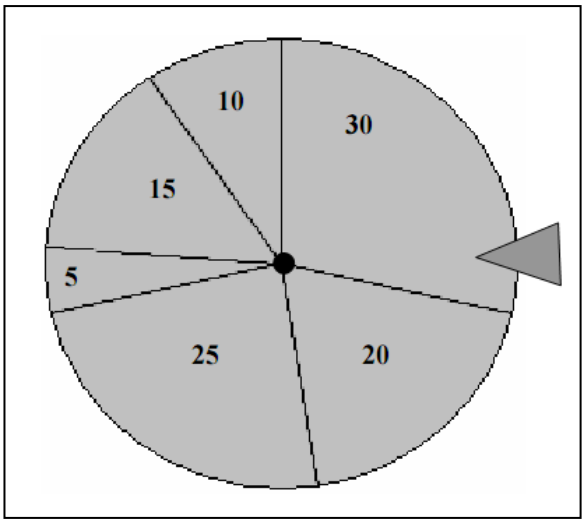


Figure 3.2. *Sélection par roulette biaisée.*

La mise en œuvre pratique se fait comme suit :

- On calcul la longueur du secteur pour chaque élément.
- On normalise la somme des longueurs à 1.
- On concatène les secteurs normalisé sur l'intervalle [0,1].
- On génère une distribution uniforme de même dimension que la taille de la population.
- Chaque élément de la distribution désignera l'élément à choisir.

En réalité puisque on utilise des populations de plus petite taille possible pour réduire le temps de calcul il est difficile d'obtenir l'espérance mathématique de cette sélection, alors un biais plus ou moins fort existe suivant la taille de la population.

- **Méthode stochastique du reste (stochastic remainder without replacement selection) :**

Elle consiste en ce qui suit :

- Pour chaque élément *i* on calcul le rapport r_i de sa fitness sur la moyenne des fitness

$$r_i = \frac{f_i}{\frac{1}{N}\sum_{i=1}^{N} f_i}$$

Soit $E[r_i]$ la partie entière de r_i , chaque élément est reproduis exactement $E[r_i]$ Fois, ce qui veut dire que tous les éléments dont le fitness est inferieur à la moyenne seront éliminés. La roue de la fortune et par suite appliquée sur les individus affectés des fitness $r_i - E[r_i]$.

Cette technique donne relativement de bons résultats pour des populations de taille réduite. La roue biaisée et ses techniques dérivées sont le plus largement utilisées, dans ce qui suit on présente quelques améliorations classiques **[24]**.

Améliorations classiques :

Les processus de sélection présentés sont très sensibles aux écarts de fitness et dans certains cas un très bon individu risque d'été produit trop souvent et peut même provoquer l'élimination complète des congénères (problème de convergence prématurée)

Pour éviter ce comportement il existe d'autres modes de Sélection (ranking) ainsi que des principes de (scaling sharing) qui empêchent les individus « forts » d'éliminer les plus faibles.

- **Scaling :**

Le scaling ou mise à l'échelle modifie les fitness à fin de réduire ou d'amplifier artificiellement les écarts de fitness entre les individus. Le processus de Sélection n'opère plus sur les fitness réelles mais sur leurs images à travers le scaling. Parmi la fonction de scaling on peut envisager un scaling linéaire ou exponentiel.

- **Le scaling linéaire :**

Dans ce cas la fonction de scaling est définie de la façon suivante :

$$f_s = a \cdot f_r + b$$

f_r : Fitness avant scaling.
f_s : Image de f_r par scaling.

En général $a < 1$ ce qui permet de réduire les écarts de fitness d'où une plus bonne exploration de l'espace de recherche.

Ce scaling ne dépend pas du numéro de la génération courante ni du nombre max de génération alors vers la fin du processus itératif on doit favoriser les bons individus **[23, 28]**.

- **Le scaling exponentiel :**

Il est défini de la façon suivante :

$$f_s = f_r^k$$

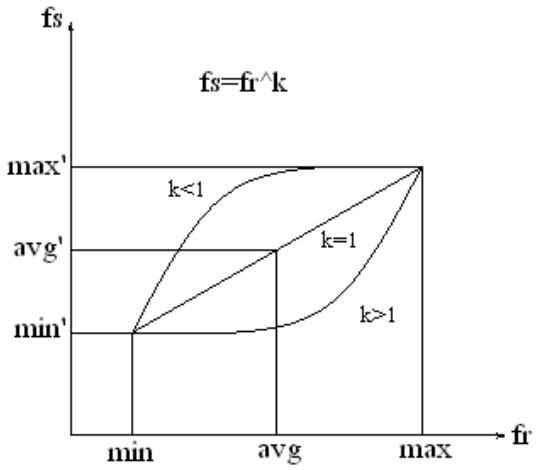


Figure 3.3. *Fonction de scaling exponentielle.*

Où n est la génération courante.

- Pour k proche de zéro on réduit fortement les écarts de fitness : aucun individu n'est vraiment favorisé et l'algorithme se comporte comme une recherche aléatoire et permet une meilleure exploration de l'espace de recherche.
- Pour k proche de un le scaling est inopérant.
- $k > 1$ les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Parmi les formules qu'on peut utiliser pour faire varier k des faibles valeurs vers les fortes au cours des générations on rencontre :

$$k = \alpha_1 \left(\tan\left(\frac{n}{N+1}\frac{\pi}{2}\right)\right) \ \ldots\ldots\ldots\ldots\ldots (3.1)$$

Où n est l'indice de la génération courante, N le nombre maximum de générations, $\alpha 1$ est un paramètre à choisir.

Cette technique s'avère bonne, mais il faut savoir qu'elle masque les modes sous optimaux.

- **Sharing :**

Dans le cas où les modes sous optimaux peuvent être intéressant on peut introduire cette technique qui consiste en la répartition sur chaque sommet de la fonction un nombre d'individus proportionnel à la fitness associée à ce sommet **[26]**.

2.4.5 Opérateur de croisement :

Le croisement a pour but l'enrichissement de la diversité de la population, en produisant de nouveaux éléments avec de nouveaux caractères.

A l'instar de son équivalent biologique, définit classiquement avec deux parents et générant deux enfants. Mais rien ne limite le nombre de parents à deux, il y a des implémentations utilisant des croisements à plus de deux parents mais ceci est rare. Classiquement le croisement associé aux chaînes de bits est le croisement par découpage de chromosome (*slicing crossover*), on rencontre dans la littérature des croisements avec un ou plusieurs points de découpage. On montre ceci dans les deux exemples suivants : le premier exemple un croisement à un point de découpage et le deuxième exemple un croisement à deux point de découpage **[34]**.

Exemple 01 :

Soit deux chromosomes chacun de M gènes. On tire aléatoirement une position de chacun des deux parents et puis en échange les deux sous-chaînes terminales de chacun des deux chromosomes ce qui produit deux enfants, comme le montre la figure suivante :

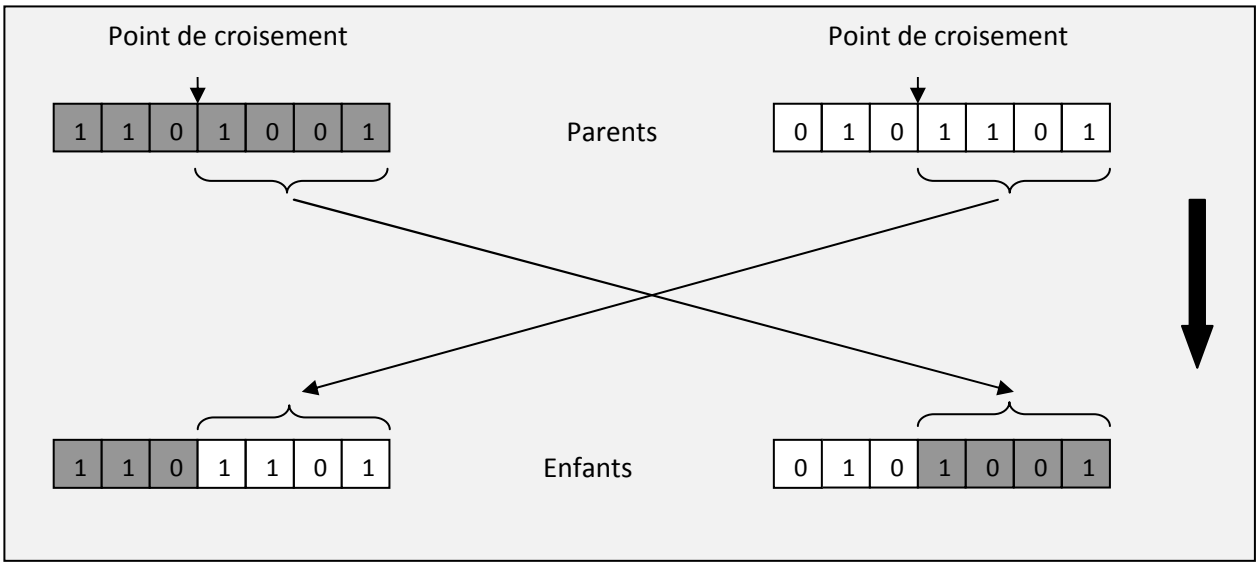


Figure 3.4. *Croisement à un seul point de découpage.*

Exemple 02 : Cet exemple montre un croisement avec deux points de découpage : la procédure étant la même :

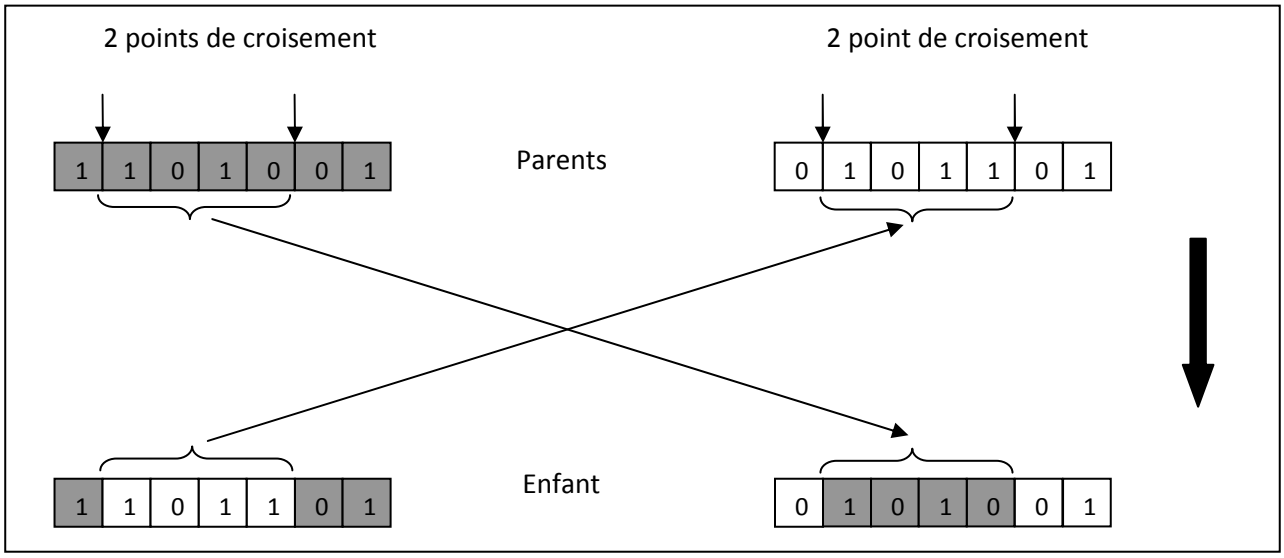


Figure 3.5. *Croisement à deux points de découpage.*

Ce type de croisement est efficace mais le fait qu'il se fait par morceaux alors il y a des gènes qui ont une très petite probabilité de croisement comme le premier gène du chromosome par exemple, en plus on peut produire des éléments non admissibles. Pour cela on a définit d'autres opérateurs de croisement beaucoup plus stochastiques alors beaucoup plus adaptés pour un algorithme heuristique.

- **Croisement uniforme :**

Dans cet opérateur on définit une nouvelle chaîne, de même longueur que les chromosomes à croiser, appelée masque. Le masque étant une chaîne de zéro et de un, on parcourt le masque si la valeur de la position i est égale à un alors les deux gènes correspondant vont se croiser sinon on les gardes tels quels, ce principe est illustré par la figure suivante **: [6]**

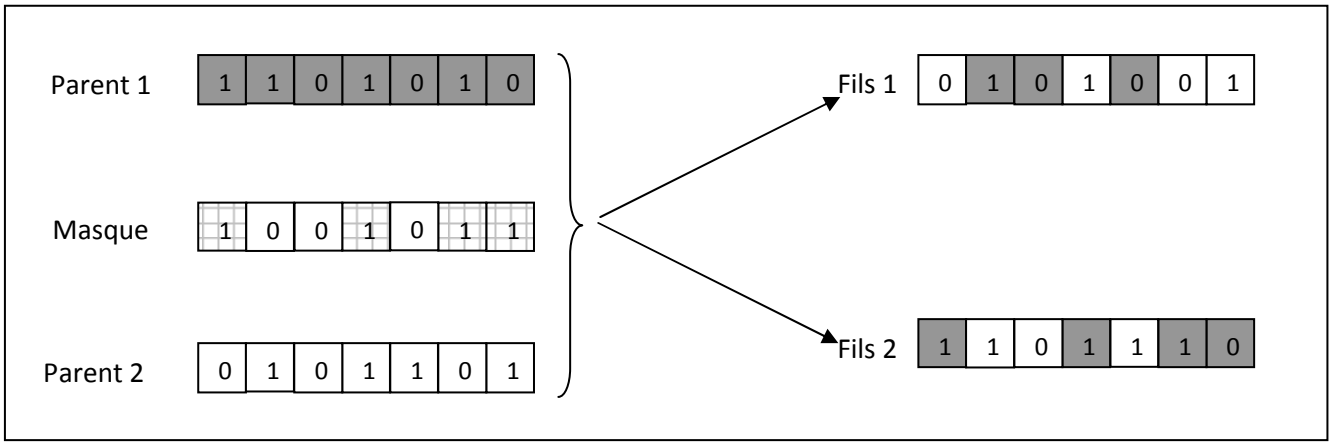


Figure 3.6. *Croisement avec masque.*

La moyenne du masque représente le taux de croisement, c'est à dire le degré de ressemblance de l'un des enfants à l'un des parents, on suppose que ce taux est maximal pour une moyenne de 1/2.

Ceci étant pour des problèmes combinatoires, de tels opérateurs pour des problèmes continus n'auront presque pas de signification, pour cela on définit des opérateurs de croisement spécifiques aux problèmes continus **[24]**.

- **Croisement barycentrique :**

Soit les deux parents $P1$ et $P2$, $P1(i)$ (respect $P2(i)$) désigne le gène i du premier (respect deuxième) parent. Soit $C1$, $C2$ leurs enfants :

$$\begin{cases} C1(i) = \alpha \cdot P1(i) + (1-\alpha)P2(i) \\ C2(i) = (1-\alpha)P1(i) + \alpha \cdot P2(i) \end{cases}$$

α Étant un coefficient de pondération adapté aux domaines d'extension des gènes, α n'est pas obligatoirement entre $[0,1]$ il peut être par exemple entre $[-0.5, 1.5]$ d'où on obtient des enfants à l'extérieur du segment $[P1(i), P2(i)]$ **[46, 47]**.

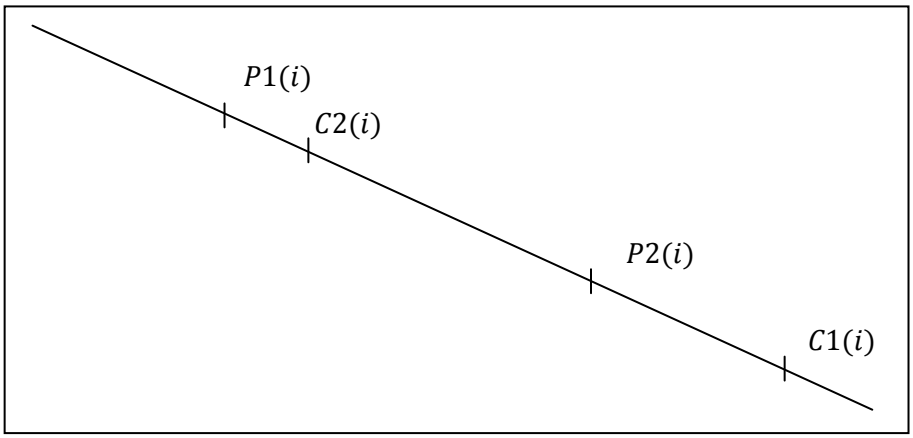


Figure 3.7. *Croisement barycentrique*.

- **Croisement avec stratégie :**

On peut par exemple adopter une stratégie de recuit pour le croisement comme on peut aussi appliquer une stratégie de jeux entre les parents et leurs enfants.

On doit noter que grâce à l'opérateur de croisement on peut générer des individus plus ou moins performants, néanmoins cet opérateur a aussi un effet destructeur car il peut donner lieu à des enfants plus mauvais que leurs parents. Pour remédier à ce problème on peut par exemple appliquer l'opérateur de croisement avec une stratégie, cependant une telle technique fragilise l'exploration de l'espace de recherche.

Dans la pratique il faut s'assurer que dans la globalité le croisement fait évoluer la population.

2.4.6 Opérateur de mutation :

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours de l'espace de recherche. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace de recherche. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement et certaines implémentations fonctionnent de cette manière. Les propriétés de convergence sont donc fortement dépendantes de cet opérateur sur le plan théorique.

L'opérateur de mutation est appliqué avec de petites probabilités $[0.01, 0.2]$, le choix des individus qui vont subir des mutations se fait de la manière suivante :

- On fixe une probabilité de mutation pm.
- On génère une distribution M, aléatoire uniforme de longueur égale à la taille de la population.
- Si $M(i) < pm$ alors l'individu i subira l'opérateur de mutation.

➢ Opérateur classique de mutation :

Définit classiquement pour les problèmes discrets, il consiste à tirer aléatoirement un gène du chromosome et le remplacé par un autre gène tiré aléatoirement aussi de l'ensemble des valeurs admissible pour ce gène ci **[25, 26]**.

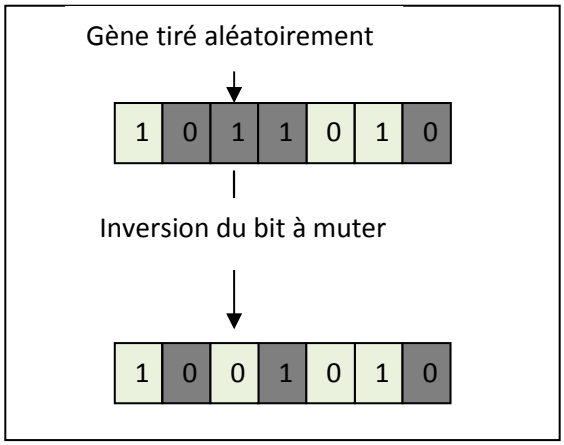


Figure 3.8. *Mutation classique (pour un codage binaire).*

➢ Opérateur de mutation pour les problèmes continus :

Pour les problèmes continus la mutation se fait un petit peu de la même manière : on tire un gène aléatoirement de l'individu à muter et on lui rajoute un bruit gaussien **[46]**.

Remarque :

À noter que les deux opérateurs ci-dessus ne dépendent ni de la longueur du chromosome ni de l'indice de la génération courante ou du nombre maximum de générations, ce qui est peu pratique.

- **Mutation adaptative :**

L'opérateur est le même que celui présenté précédemment, la différence est que pour cet opérateur l'amplitude du bruit gaussien à rajouté dépend de l'indice de la génération actuelle ainsi que le nombre maximum de génération. Il y a plusieurs façons pour faire varier cette amplitude, et cela dépend fortement du problème à traiter.

L'un des majeurs inconvénients des algorithmes génétiques est leur mauvaise convergence locale. Il est possible d'affiner cette convergence en imposant que l'amplitude des mutations soit décroissante au cours des générations, et cela pour les problèmes codés réels. Par exemple l'amplitude de la mutation (l'amplitude du bruit gaussien) à la génération $i+1$ est l'écart moyen entre la fitness du meilleur individu et les fitness du reste de la population de la génération i **[46]**.

3. Algorithmes génétiques et logique floue :

3.1 Introduction :

La synergie entre les algorithmes génétiques et la logique floue peut apparaitre dans trois formes complémentaires.

La forme la plus directe exploite la capacité de l'algorithme à rechercher des optimums pour la synthèse de contrôleurs flous. C'est le cas de notre travail.

Pour la seconde forme, les algorithmes génétiques sont relativement faciles à implémenter et leur performance en général tend à devenir de plus en plus satisfaisante comparé au peu de connaissance qu'on a à propos du problème sur lequel on les appliquera ; néanmoins en pratique ils requièrent une sorte de supervision humaine pendant leur utilisation. L'idée alors est d'utiliser une connaissance linguistique (floue) pour détecter l'émergence d'une solution, et adapter les paramètres de l'algorithme, et éviter à l'algorithme les comportements non désirés tel que la convergence prématurée.

Une troisième option est d'hybrider les techniques de l'arithmétique floue dans l'algorithme génétique lui-même. Pour l'instant quelques précisons sur le calcul du *fitness* peuvent être sacrifié en définissant une *fitness* floue, ou éventuellement des allèles floue pour les gènes, cela *fuzzifiera* alors même les opérateurs génétiques.

Dans ce qui suit on détaillera les deux dernières formes, vu que la première forme est l'objet de notre travail et elle est très bien détaillée par la suite.

3.2 Contrôle flou de l'évolution :

L'adaptation des paramètres de l'algorithme génétique est la clé pour la détermination du compromis exploration/exploitation. Il est bien connu que l'ajustement de paramètres a un impact significatif sur les performances de l'algorithme. Souvent un mauvais choix des paramètres peut causer une convergence très lente ou une convergence prématurée. Toutefois la détermination de paramètres robustes pour tous les problèmes est difficile voire impossible, ceci puisque l'interaction paramètres comportement de l'algorithme est très

complexe, et les paramètres optimaux sont dépendants du problème à résoudre. En outre, différent paramètres peuvent être optimaux à des instant différent du même processus d'optimisation.

Pour ces raisons, un algorithme génétique nécessite une certaine supervision pendant son utilisation pour la résolution de problèmes pratiques. Un superviseur peut agir et modifier les paramètres durant le processus d'optimisation jusqu'à l'obtention d'un comportement souhaité. Finalement, il y a aussi le problème de la décision à l'arrêt de l'algorithme, le critère d'arrêt. Ce dernier problème est résolu en fonction de la satisfaction du résultat trouvé ainsi que les contraintes sur le temps, bien sûr, c'est une intervention du superviseur.

L'alternative au superviseur humain est une adaptation automatique des paramètres de l'algorithme, capable de s'auto ajuster pour atteindre les meilleurs performances. En remarquant que l'algorithme génétique a une dynamique complexe fortement non linéaire, on aperçoit rapidement qu'il peut tirer un avantage énorme d'une technique aussi forte que le control par logique floue.

3.2.1 Fuzzy governement :

Le nom suggéré *fuzzy governement* peut être attribué à la collection de règles floues chargées de contrôler l'évolution d'une population, détecter l'émergence d'une solution, adapter les paramètres de l'algorithme *on flight* et éviter les comportements indésirables.

Un contrôleur flou implémente un opérateur expert pour la prise de décision. Le *fuzzy governement* adapte les paramètres de l'algorithme ou les opérateurs génétiques dans l'ordre d'assurer le meilleur rapport exploitation/exploration. Le nom de *fuzzy governement* est motivé par la similarité avec un gouverneur de nation qui, au moins en principe, exploite son pouvoir limité pour assurer les meilleures conditions de vie et de prospérité pour son peuple.

L'attractivité à l'utilisation de supervision floue est qu'elle donne la possibilité d'inclure des connaissances générales et imprécises du savoir faire du développeur de l'algorithme, d'une manière simple et directe.

3.2.2 Architecture d'un algorithme génétique adaptatif :

L'idée principale, d'une version adaptative d'un algorithme génétique, représentée sur la ***figure 3.9***, est d'utiliser un contrôleur flou dont les entrées sont des statistiques sur l'évolution de l'algorithme et les sorties sont les paramètres à ajuster.

Les statistiques sont acquises de l'algorithme à un certain pas d'échantillonnage, soit une fois par r générations mais jamais plus d'une fois par génération, et elles sont envoyés au superviseur flou, alors suivant les règles de décision de nouveau paramètres sont calculés, transmis à l'algorithme et la main rendu au processus d'optimisation.

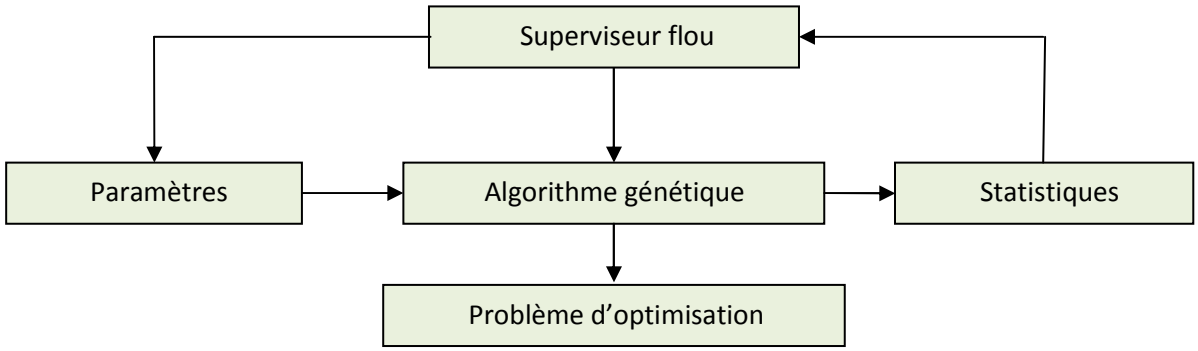


Figure 3.9. *Architecture d'un algorithme génétique adaptatif.*

L'encombrement apporté par le superviseur tout fois est négligeable, et il peut être réduit d'avantage en réduisant la fréquence d'adaptation $1/r$. En contre partie l'expérience a montré qu'on peut avoir une amélioration considérable des performances **[27]**.

3.2.3 Les Statistiques :

Les statistiques d'un algorithme génétique peuvent principalement être divisées en deux classes : les statistiques génotypiques, et phénotypiques.

Les mesures de diversité constituent un type de statistiques génotypiques. Elles sont basées sur la définition d'une mesure de similarité floue.

$$\mu_{similar}(\gamma\ ,\kappa)$$

Où $\gamma\ et\ \kappa$ sont deux génotypes, d'autres mesures génotypiques peuvent être définies d'une manière plus ou moins *ad hoc*.

Les mesures de diversité phénotypiques portent sur les performances des individus, telles que, rang du fitness, meilleur et moyenne fitness ou leurs rapport, la variance du fitness ou on minimum.

On peut définir d'autres mesures comme le taux de succès qui est le nombre de cas où la mutation aboutit à un individu meilleur sur le nombre total des mutations **[27]**, comme on peut aussi dans un cadre u peu plus complexe de tracer les trajectoires des solutions émergentes.

3.2.4 La table de règles :

Différentes approches pour la construction de la table de décision sont disponible dans la littérature. Une alternative est l'étude du comportement dynamique de l'algorithme pour gagner une expertise sur le problème, puis cette expertise couplée avec les connaissances générales sur les algorithmes génétiques seront formulées sous la forme de règles générales et imprécises.

3.2.5 Exploitation de l'expertise :

Un superviseur flou a été conçu pour résoudre les deux problèmes de convergence prématurée et de la convergence très lentes observées pout tout les problèmes de l'optimisation. Ces problèmes sont dus au moins au trois facteurs :

1. Les paramètres ne sont pas bien choisis initialement pour une tache donnée.
2. Les paramètres sont constants au cours du temps bien que la condition d'évolution est dynamique.
3. L'interaction entre les différents paramètres complexe et difficile à comprendre.

Un superviseur flou peut aider dans les trois situations. Il peut être utilisé pour choisir les paramètres de l'algorithme même avant son lancement, pour leur adaptation on line comme il peut être utilisé pour assister un développeur afin de développer, d'implémenter et de valider un algorithme génétique dédiée à une tache donnée. Le ***tableau 01*** donne un exemple de règles appliquées pour ajuster les taux de croisement et de mutation d'un algorithme génétique.

p_c	Taille de la population		
Génération	petite	moyenne	grande
petite	moyenne	petite	petite
moyenne	grande	grande	moyenne
grande	Très grande	Très grande	grande

p_m	Taille de la population		
Génération	petite	moyenne	grande
petite	grande	moyenne	petite
moyenne	moyenne	petite	Très petite
grande	petite	Très petite	Très petite

Tableau 01. *La base de règles pour l'adaptation des taux de croisement p_c, et de mutation p_m, respectivement* ***[27]****.*

Un algorithme génétique adaptatif est donné où le superviseur flou ajuste deux paramètres p_e et η_{min} . Le superviseur modifie le taux d'application d'un opérateur de croisement orienté exploitation, qui est utilisé ainsi qu'un autre opérateur de croisement conventionnel orienté exploration. Quand p_e est petit l'opérateur génère une diversité et le critère exploration est favorisé, avec l'accroissement de p_e l'information génétique existante est exploitée pour générer de meilleurs individus. Le paramètre η_{min} caractérise la sélectivité ou pression de sélection, avec ce paramètre on réalise au comportement désiré exploration/exploitation. Les diversités génotypique ainsi que phénotypique seront les

entrée du superviseur flou, les sortie sont Δp_e et $\Delta \eta_{min}$ les deux compris dans l'intervalle $[\frac{1}{2}\ ;\frac{3}{2}]$, qui détermine la variation de p_e et η_{min} comme suit :

$$p_e \quad \leftarrow \quad p_e \Delta p_e\ ;$$

$$\eta_{min} \quad \leftarrow \quad \eta_{min} \Delta \eta_{min}\ ;$$

La table de règles est donnée comme suit :

Δp_e	Diversité phénotypique		
Diversité génotypique	basse	moyenne	haute
basse	moyenne	petite	petite
moyenne	grande	grande	moyenne
haute	grande	grande	moyenne

$\Delta \eta_{min}$	Diversité phénotypique		
Diversité génotypique	basse	moyenne	haute
basse	petite	moyenne	grande
moyenne	petite	grande	grande
haute	petite	petite	grande

Tableau 02. *Table de règles pour l'adaptation du taux de croisement orienté exploitation/exploration p_e, et la pression de sélection η_{min} , respectivement* ***[27]****.*

A partir de ces deux exemples on remarque que l'exploitation des techniques de la logique floue pour la supervision de l'évolution d'un algorithme génétique semble très intéressante et peut considérablement améliorer les performances de l'algorithme.

L'application des algorithmes génétiques pour la résolution de problèmes d'optimisation pratique reste une tache complexe et des fois même difficile à réaliser vu que les performances de l'algorithme sont *problem dependent*, ce qui veut dire que pour chaque problème plusieurs tests et modifications de l'algorithme seront réalisés ce qui n'est pas toujours possible, par exemple dans le cas où la fonction à optimiser est le résultat d'une simulation qui prend énormément du temps (tel est le cas dans notre application qu'on détaillera plus tard).

4. Algorithmes génétiques et calcul parallèle :

L'intérêt de la parallélisation des algorithmes génétiques est de gagner en temps de calcul. Il existe pour cela au moins deux méthodes utilisées classiquement :

4.1 Les îles :

La première consiste à diviser la population de taille n en N sous population et à les répartir sur l'ensemble des machines dont on dispose.

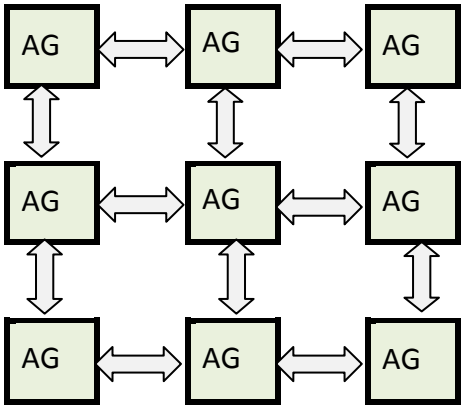


Figure 3.10. *Architecture parallèle distribuée.*

4.2 Le style maitre-esclave :

La seconde maintient la population totale sur une seule machine mais se sert des autres pour y envoyer les évaluations afin qu'elles se fassent en même temps.

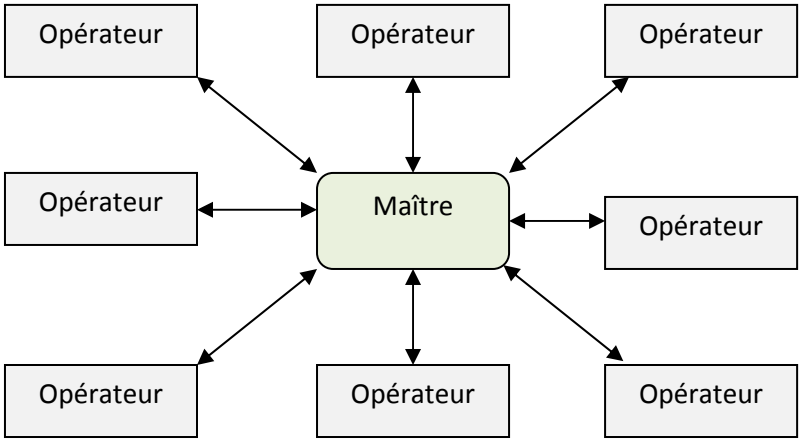


Figure 3.11. *Architecture Maitre-Esclave.*

5. Conclusion :

Dans ce chapitre nous avons présenté les bases des algorithmes génétiques ainsi que quelques outils pratiques pour l'implémentation des différents opérateurs et étapes. Nous avons vu que l'efficacité d'un algorithme génétique dépend fortement du codage du problème et des opérateurs utilisés, malheureusement il n'existe pas de méthodes systématiques pour la détermination de ces derniers, et les performances de l'algorithme dépendent du problème à résoudre.

Nous avons aussi introduit la notion d'algorithmes génétiques flous, ainsi que l'implémentation parallèle des ces derniers.

Chapitre IV :

Optimisation par Essaims Particulaires

1. Introduction :

L'optimisation par essaims particulaires (***OEP***) est une métaheuristique d'optimisation, inventée par ***Russel Eberhart*** (ingénieur en électricité) et ***James Kennedy*** (socio-psychologue) en 1995.

Cette méthode s'inspire à l'origine du monde du vivant. Elle s'appuie notamment sur un modèle développé par le biologiste *Craig Reynolds* à la fin des années quatre vingt, permettant de simuler le déplacement d'un essaim (inspirée du comportement social des animaux évoluant en essaim). Une autre source d'inspiration, revendiquée par les auteurs, est la socio-psychologie.

Cette méthode d'optimisation se base sur la collaboration des individus entre eux. Elle est d'ailleurs une méthode récente parmi les algorithmes évolutionnaires, qui s'appuient eux sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus peu intelligents peut posséder une organisation globale complexe.

L'échange d'information entre eux fait que, globalement, ils arrivent néanmoins à résoudre des problèmes difficiles, comme c'est le cas, par exemple, chez les abeilles vivant en essaim (exploitation de sources de nourriture, construction de rayons, etc.)

Ainsi, grâce à des règles de déplacement très simples (dans l'espace des solutions), les particules peuvent converger progressivement vers un minimum global. Cette métaheuristique semble cependant mieux fonctionner pour des espaces en variables continues.

Donc l'optimisation par essaim particulaire (OEP), dans sa version historique, est une méthode itérative collective, anarchique au sens originel du terme, mettant l'accent sur la coopération, partiellement aléatoire et sans sélection. Ca sera l'objet de cette partie de ce chapitre que de détailler ces caractéristiques et de les formaliser pour obtenir un modèle exploitable, particulièrement efficace pour les problèmes fortement non linéaires.

2. Origines et métaphore :

Vers 1927, Karl Von Frish a découvert que les abeilles ramenaient à la ruche non seulement du nectar et du pollen, mais aussi de l'information… Il a patiemment décodé leur langage et l'observateur attentif peut maintenant les comprendre partiellement.

L'optimisation par essaim particulaire (OEP) est une méthode née aux Etats-Unis sous le nom de ***Particle Swarm Optimization*** (***PSO***).

Initialement, les deux concepteurs, Russel Eberhart et James Kennedy, cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information.

La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations seulement des connaissances locales. Un modèle simple fut alors élaboré.

Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants convergeant parfois en plusieurs sous-essaims vers des sites intéressants. Ce comportement se retrouve dans bien d'autres modèles, explicitement inspirés des systèmes naturels. Ici, la métaphore la plus pertinente est probablement celle de l'essaim d'abeilles, particulièrement du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ses consœurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement c'est-à-dire une abeille doit parfois combiner plusieurs informations, celles correspondant à sa propre connaissance du terrain et celles apportées par une butineuse, voire plusieurs presque simultanément. La manière dont elle le fait reste un mystère mais pour nous inspirer du comportement de nos abeilles, il nous faudra quand même la modéliser, donc, inventer une méthode de toute pièce. Le modèle s'est révélé être trop simple pour vraiment simuler un comportement social, mais par contre très efficace en tant qu'outil d'optimisation **[38]**.

Figure 4.1 : Etudes du comportement des abeilles dans une ruche (teste sur 5000 abeilles grâce à un repérage par des étiquettes numérotées) **[40]**.

Comme nous allons le voir, le fonctionnement de l'OEP fait qu'elle peut être rangée dans les méthodes itératives (on approche peu à peu de la solution) et stochastiques (on fait appel au hasard). Sous ce terme un peu technique, on retrouve un comportement qui est aussi vieux que la vie elle-même : améliorer sa situation en se déplaçant partiellement au hasard et partiellement selon des règles prédéfinies.

3. Formulation :

En OEP, un « site intéressant » correspond à un optimum au moins local d'une certaine fonction définie dans un espace de recherche. Cette fonction peut être donnée par une formule mathématique ou, à défaut, par un algorithme, voire par le résultat du déroulement d'un processus, réel ou simulé. Le tout est que l'on sache calculer sa valeur en chaque point.

Pour sa version classique l'OEP cherche les sites les plus intéressants c'est-à-dire l'optimum global de notre fonction fitness, pour ce faire, l'OEP s'inspire du comportement coopératif décrit dans notre métaphore : chaque particule est capable de communiquer à certaines autres la position et la qualité du meilleur site qu'elle connait, qualité que l'on peut interpréter comme étant sa 'valeur'.

3.1 Taille de l'essaim :

Tout d'abord, il faut définir un essaim dans l'espace de recherche, mais de quelle taille ? Les véritables essaims d'abeilles comptent typiquement 20000 individus, nous nous contenterons de taille de l'ordre de 20 à 40. Il s'avère en effet qu'en OEP ces tailles sont très souvent suffisantes.

Ce qui compte plutôt c'est le nombre de fois où la fonction fitness doit être évaluée car dans la plupart des problèmes réels, cette évaluation nécessite un temps non négligeable, et évidement, pour une itération, ce nombre d'évaluation est égal au nombre de particules. Donc si nous voulons réduire le nombre total d'évaluations nécessaires pour trouver une solution, nous somme au contraire tentés de diminuer la taille de l'essaim. Mais un essaim trop petit risque de mettre très longtemps pour trouver une solution ou même ne pas la trouvé du tout.

Donc il ya un compromis à trouver. Empiriquement, les expérimentateurs ont proposé des tailles de l'ordre de 20 à 30 particules qui, en effet, se révèlent tout à fait suffisante pour résoudre la quasi-totalité des problèmes de test classiques **[36]**.

3.2 Liens d'information :

Il faut définir, pour chaque particule, qui sont ses informatrices. Toujours par analogie avec ce qui ce passe (apparemment) dans une ruche, nous pouvons définir au hasard pour chaque particule son groupe d'informées, ce qui, automatiquement, détermine également les informatrices de chaque particule, puisque, formellement, nous établissons un graphe de relation entre les particules.

Combien d'informées, combien d'informatrices ? D'une part, si toutes les particules sont informées par chacune, toute l'information acquise à chaque instant est diffusée immédiatement, ce qui semble favorable. Mais d'autre part le risque est grand d'avoir un comportement trop uniforme : avec les mêmes informations, les particules vont agir de la même manière. Pour les recherches difficiles ce n'est pas efficace. Inversement, si chaque

particule n'a que trop peu d'informatrices, nous pourrons obtenir des comportements plus diversifiés, mais le risque est alors que l'information soit mal transmise. Or il est important que si une particule trouve un bon emplacement, toutes les autres, plus ou moins directement, puissent avoir connaissance de cette information, pour en tirer parti.

Dans ces conditions, comme nous l'avons vu, si le choix est fait au hasard à chaque pas de temps, prendre deux ou trois informées pour chaque particule semble un bon compromis, il ya aussi d'autres versions de l'OEP qui ne fait pas ce choix au hasard mais selon une règle tenant compte de nos deux critères, par exemple une sélection automatique permanente et judicieuse des informatrices.

La nature des informations transmises est évidemment importante, mais plus il y aura d'informations, plus il sera long et difficile à une particule de les traiter. Le plus délicat à modéliser est la manière dont une particule informée va calculer son prochain déplacement. Tout d'abord, notons qu'elle est en général déjà entrain de se déplacer : elle possède donc une certaine vitesse. Ensuite, puisqu'elle est informatrice éventuelle d'autres particules, elle connait sa propre meilleure performance. Enfin, donc, elle connait toutes les meilleures performances de ces informatrices. Simplifions : ne gardons que la meilleure.

Il nous reste donc trois éléments à combiner : vitesse propre, meilleure performance propre et meilleure des meilleures performances des informatrices.

Imaginons trois cas extrême : premier cas, la particule est aventureuse et n'entend suivre que sa propre voie, alors elle va attribuer une confiance nulle aux informations reçues et même à sa propre expérience ; elle se contentera de suivre plus au moins la direction déjà suivie, c'est-à-dire que le prochain déplacement se fera en gros avec la même vitesse (intensité et direction) que le précédent. Deuxième cas, elle est très conservatrice ; elle va accorder une grande confiance à sa meilleure performance et tendra à y revenir sans cesse. Troisième cas, elle ne s'accorde à elle-même aucune confiance ; elle se déplacera selon les indications de sa meilleure informatrice **[36]**.

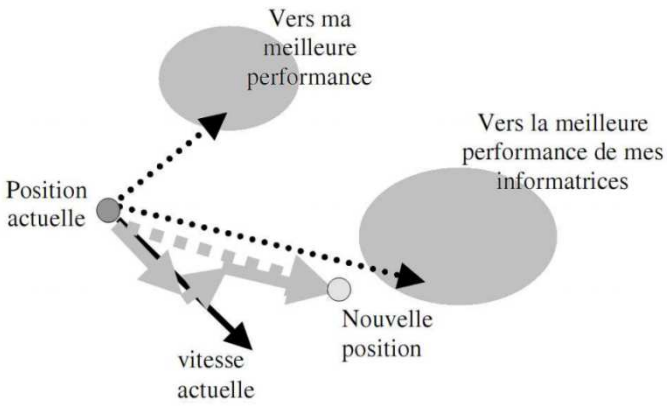


Figure 4.2 : *les trois éléments fondamentaux pour le calcul du prochain déplacement d'une particule.*

Nous avons donc trois déplacements fondamentaux, schématisés par la figure précédente : selon sa vitesse actuelle, vers sa propre meilleure performance et vers celle de sa meilleurs informatrice. Pour calculer le déplacement vrai à partir de ces trois vecteurs de base, le plus simple est d'en faire une pondération linéaire, grâce à des ***coefficients de confiance***. Tout l'art des premières versions d'OEP consiste en la définition judicieuse de ces coefficients.

3.3 Initialisation:

L'initialisation consiste simplement à placer d'abord au hasard les particules selon une distribution uniforme dans l'espace de recherche. Ceci est une étape que l'on retrouve dans à peu prés tous les algorithmes d'optimisation itérative stochastique.

Mais ici, en plus, les particules possèdent des vitesses. Par définition une vitesse est un vecteur ou, plus précisément, un opérateur, qui, appliqué à une position, va donner une autre position. C'est donc en fait un déplacement, appelé vitesse car le pas de temps des itérations est toujours implicitement considéré comme égale à 1.

En pratique, il n'est pas souhaitable que trop de particules tendent à sortir de l'espace de recherche dés le premier pas du temps, ni d'ailleurs plus tard. Pour les premières formulations nous contentons de tirer au hasard les valeurs des composantes de chaque vitesse, selon une distribution uniforme dans :

$$[(x_{min} - x_{max})/2\ , (x_{max} - x_{min})/2\ ]$$

3.4 Equation de mouvement :

Une caractéristique très importante de l'OEP du moins dans ses versions classiques est qu'elle ne pratique aucune sélection. L'idée est en effet que les faibles d'aujourd'hui sont peut être les forts de demain. Les particules à piètre performance sont conservées, avec l'espoir que parmi elles se trouvent précisément les originales, les dissidentes qui permettront de découvrir le meilleur site existant dans l'espace de recherche. Les expérimentations ont d'ailleurs parfaitement justifié cet espoir **[36]**.

L'espace de recherche est de dimension ***D***. la position courante d'une particule dans cet espace à l'instant ***t*** est donnée par un vecteur ***x(t),*** à ***D*** composantes, donc. Sa vitesse courante est ***v(t)***. La meilleure position trouvée jusqu'ici par cette particule est donnée par un vecteur ***p(t)***. Enfin, la meilleure position trouvée par les informatrices de la particule est indiquée par un vecteur ***g(t)***. En général, nous écrivons simplement ***x, v, p, g***. La $d^{ième}$ composante de l'un quelconque de ces vecteurs est indiquée par l'indice ***d*** par exemple x_d. Avec ces notations les équations de mouvement d'une particule sont, pour chaque dimension d :

$$\begin{cases} v_d \leftarrow c_1 v_d + c_2(p_d - x_d) + c_3(\mathrm{g}_d - x_d) \\ x_d \leftarrow x_d + v_d \end{cases} \quad (1)$$

Les coefficients de confiances sont définis de la manière suivante :

- c_1 est constant (confiance en son propre mouvement)
- c_2 et c_3 (respectivement confiance en sa meilleure performance et en celle de sa meilleure informatrice) sont choisis au hasard à chaque pas de temps selon une distribution uniforme dans un intervalle $[0, c_{max}]$ donné.

C'est pourquoi le système **(1)** peut être réécrit de manière plus explicite, en mettant en évidence les systèmes aléatoires les éléments aléatoires :

$$\begin{cases} v_d \leftarrow c_1 v_d + c_{max} alea(0,1)(p_d - x_d) + c_{max} alea(0,1)(\mathrm{g}_d - x_d) \\ x_d \leftarrow x_d + v_d \end{cases} \quad (2)$$

Pour utiliser ce modèle, on doit définir les deux paramètres c_1 et c_{max} , ce dernier peut être vu comme la confiance maximale accordée par la particule toute performance transmise par une autre. Pour chaque problème les «bonnes» valeurs ne peuvent être trouvées qu'expérimentalement, à l'aide cependant de deux règles empiriques, dégagées après de nombreux tests.

La première règle stipule que c_1 doit être de valeur absolue inférieure à 1. Elle se comprend intuitivement si l'on considère ce qui se passe lors de plusieurs pas de temps successifs, dans le cas très particulier ou la particule est et reste elle même sa meilleure informatrice. Nous avons alors en effet $p_d = x_d = \mathrm{g}_d$ et à chaque pas de temps la vitesse est simplement multipliée par c_1. S'il est de valeur absolue supérieure à 1, la vitesse augmente sans cesse et la convergence est impossible. Notons que en théorie rien n'interdit à ce coefficient d'être négatif ; le comportement obtenu étant alors fortement oscillatoire, mais ce n'est jamais le cas en OEP classique. Nous le supposerons donc positif.

En pratique ce coefficient ne doit pas être ni trop petit, ce qui induit une convergence prématurée, ni trop grand, ce qui, au contraire, peut ralentir exagérément la convergence. Les auteurs des premiers travaux sur l'OEP préconisaient de la prendre égale à 0,7 ou 0,8.

La deuxième règle indique simplement que le paramètre c_{max} ne doit pas être trop grand, une valeur de l'ordre de 1,5 à 1,7 étant considérée comme efficace dans la plupart des cas. A l'époque ou elle a été énoncée, cette règle n'avait pas de justification, même intuitive. Elle était purement expérimentale.

En effet, les valeurs préconisées sont très proches de celles déduites plus tard d'analyses mathématiques montrant que pour une bonne convergence les valeurs de c_1 et

c_{max} ne doivent pas être choisies indépendamment **[36] [38]** , par exemple les couples de valeurs (0,7 1,47) et (0,8 1,62) sont effectivement corrects. Les premiers expérimentateurs ; *James Kennedy et Russel Eberhart*, auxquels on peut ajouter *Yuhui Shi,* ont fait un bon travail.

3.5 Confinement d'intervalle :

Lors des premières expérimentations de l'OEP les fonctions utilisées étaient définies pour toutes valeurs, lors de l'évolution de l'essaim il pouvait arriver qu'une particule sorte de l'espace de recherche initialement défini, mais cela était sans importance puisque la valeur de sa position pouvait en fait encore être calculée, sans planter l'exécution. Néanmoins, évidement cela n'est pas toujours le cas. Par exemple, dans la plupart des langages de programmation et avec la plupart des compilateurs, l'évaluation d'une fonction telle que :

$$f(x) = \sum_{d=1}^{D} \sqrt{x_d}$$

renvoie un message d'erreur dès que l'une des coordonnées x_d est négative.

Par conséquent, il a fallu très vite ajouter un mécanisme pour éviter qu'une particule sorte de l'espace de recherche. Le plus simple de tous est le confinement d'intervalle. Supposons toujours, par simplicité, que l'espace de recherche soit $[x_{min}, x_{max}]^D$ alors ce mécanisme stipule que si une coordonnées xd calculée selon les équations de mouvement sort de l'intervalle $[x_{min}, x_{max}]$ on lui attribue en fait la valeur du point frontière le plus proche. En pratique cela revient à remplacer la deuxième ligne du système **(1)** par :

$$x_d \leftarrow Min(Max(x_d + v_d, x_{min}), x_{\max})$$

Cette forme simple, quoique donnant des résultats corrects, a cependant un inconvénient. En effet, nous sommes dans un cas de figure ou la vitesse propre de la particule tend à la faire sortir de l'espace de recherche. Le confinement précédent ramène certes la particule à la frontière de l'espace de recherche, mais ne change pas la vitesse. Celle-ci est recalculée et donc en général modifiée au prochain de temps, mais il n'est pas rare qu'elle reste plus ou moins orientée de la même manière, alors la particule tendra à nouveau à dépasser la frontière, y sera ramenée par le confinement, etc. en pratique, tout se passera comme si elle était « collée » à cette frontière.

C'est pourquoi on doit compléter le mécanisme de confinement par une modification de la vitesse. On peut remplacer la composante qui pose problème par son opposé, éventuellement pondéré par un coefficient inférieur à 1, ou encore, tout simplement, l'annuler. Si l'on veut choisit l'annulation, le mécanisme complet est alors décrit par les opérations suivantes : **[36] [38]**

$$x_d \notin [x_{min}, x_{max}] \Longrightarrow \begin{cases} v_d \leftarrow 0 \\ x_d < x_{min} \Longrightarrow x_d \leftarrow x_{min} \\ x_d > x_{max} \Longrightarrow x_d \leftarrow x_{max} \end{cases}$$

L'adaptation ou cas ou les intervalles définissant l'espace de recherche sont différents pour chaque dimension est immédiate, mais ce qu'il faut surtout retenir, c'est le principe même du confinement, qui stipule ceci : « si une particule tend à sortir de l'espace de recherche, alors la ramener au point le plus proche qui soit dans cet espace et modifier sa vitesse en conséquence ».

3.6 Structure de l'Algorithme :

L'OEP est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, chaque particule est déplacée suivant les équations de mouvement données précédemment (1). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les $\boldsymbol{p_i}$ ainsi que $\boldsymbol{g}$ sont alors mis à jour. Cette procédure est résumée par l'Algorithme suivant. ***N*** est le nombre de particules de l'essaim.

Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimum global est connu a priori, on peut définir une « erreur acceptable » $\boldsymbol{\varepsilon}$ comme critère d'arrêt. Sinon, il est commun de fixer un nombre maximum d'évaluations de la fonction objectif ou un nombre maximum d'itérations comme critère d'arrêt. Cependant, au regard du problème posé et des exigences de l'utilisateur, d'autres critères d'arrêt peuvent être utilisés.

❖ ***Algorithme d'optimisation par essaim particulaire***

Initialisation aléatoire des positions et des vitesses de chaque particule
Pour chaque particule i, $p_i = x_i$
Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour *i = 1 à N* **faire**
    Déplacement de la particule

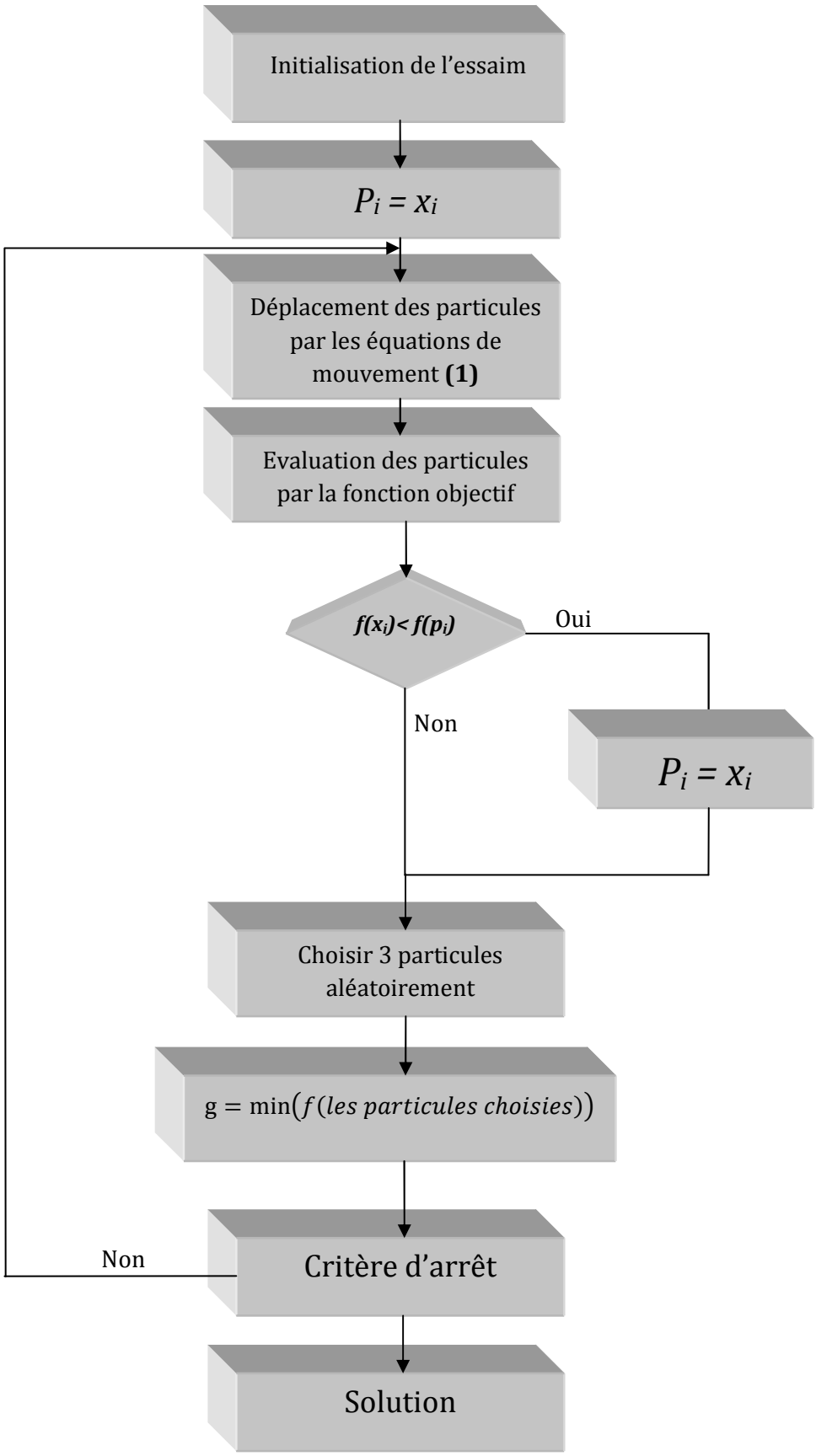
$$\begin{cases} v_d \leftarrow c_1 v_d + c_2(p_d - x_d) + c_3(\mathrm{g}_d - x_d) \\ x_d \leftarrow x_d + v_d \end{cases}$$

    Évaluation des positions
      Si $f(x_i) < f(p_i)$
        $p_i = x_i$
      Fin Si
    Groupe d'informatrices = {choisir 3 particules aléatoirement}

$$\mathrm{g} = \min\big(f(groupe\ d'informatrices)\big)$$

  Fin Pour
Fin Tant que

❖ ***Organigramme :***



4. Deux erreurs courantes :

Les équations de mouvement sont parfois décrites sous forme vectorielle :

$$\begin{cases} v \leftarrow c_1 v + alea(0, c_{max})(p - x) + alea(0, c_{max})(\mathrm{g} - x) \\ x \leftarrow x + v \end{cases}$$

Dans ce cas, conformément à la définition de la multiplication d'un vecteur par un coefficient, cela signifie que toutes les composantes, sont multipliées par le même nombre aléatoire. C'est une erreur au sens ou cela correspond à un algorithme différent de celui de l'OEP mais nous pouvons considérer cette forme comme une variante. Il faut cependant noter que les meilleurs paramétrages pour c_1 et c_{max} passent par l'utilisation d'un coefficient de constriction et que cette variante est beaucoup moins efficace que la forme classique.

La proximité de ***p*** (resp. ***g***) est ici un simple segment et la distribution des possibles pour le prochain déplacement est un D-parallélépipède situé « entre » ***p*** et ***g*** ce qui restreint l'exploration, en particulier parce que tout un ensemble de points situé prés de ***p*** (resp. ***g***) n'a aucune chance d'être choisi.

L'autre erreur consiste à effectuer une mise en facteur dans la première équation de mouvement

$$v_d \leftarrow c_1 v_d + alea(0, c_{max})(p_d + \mathrm{g}_d - 2x_d)$$

Ou encore

$$v_d \leftarrow c_1 v_d + alea(0, 2c_{max}) \left(\frac{p_d + \mathrm{g}_d - 2x_d}{2} \right)$$

Sous cette forme, nous voyons que la prochaine position sera alors prise au hasard selon une distribution uniforme dans un hyperparallélépipède dont l'arrête pour la dimension ***d*** est de longueur :

$c_{max}\,|pd + gd|$ et dont le centre se trouve en ajoutant au vecteur x le vecteur $c_1\, v\, (\, p\, +\, g\,)/2$**.** A vrai dire on pourrait simplement parler de variante plutôt que d'erreur, car cette distribution est presque aussi riche que celle d'origine.

5. L'essaim – mémoire :

Depuis le début nous avons postulé que chaque particule était apte à remplir ces deux fonctions. Cela nous a conduit à des formulations assez alambiquées, comme « la meilleure des meilleures positions trouvées jusque maintenant par les informatrices ». De plus l'information mémorisée par la particule elle – même est traitée séparément de celle apportée par les autres, alors qu'en fait rien ne les distingue dans leur nature, également cela impose de mémoriser autant de positions que de particules, ni plus, ni moins. Enfin il

pourrait être souhaitable de mettre directement en relation les positions mémorisées pour en déduire de nouveaux déplacements intéressants. On peut certes quand même étudier diverses topologies, mais sans toute la souplesse souhaitable.

C'est pourquoi il est intéressant de changer légèrement de point de vue et de considérer que les fonctions d'exploration et de mémorisation sont portées par des particules distinctes, cela nous permettra aussi de définir plus facilement différentes sortes de groupes d'informatrices.

Ainsi nous aurons comme avant un *essai – explorateur*, composé par particules turbulentes, se déplaçant à chaque pas de temps, mais en plus aussi un essaim – mémoire. Ses particules, sont appelées *mémoires* et que nous pouvons imaginer lourdes, lentes et sages ne se déplacent qu'occasionnellement et seulement à coup sûr vers les meilleures positions signalées par les exploratrices, ainsi, l'association d'une mémoire et d'une exploratrice correspond à une particule selon la terminologie historique.

6. Paramétrages optimaux :

Optimiser l'optimiseur ! , c'est un problème d'optimisation du second niveau, en effet, considérons une fonction ***f*** à minimiser par notre algorithme d'OEP paramétrique, dans nos exemples le critère d'arrêt est un nombre donné ***T*** d'évaluations de ***f***, les étapes sont les suivantes :

a) Choisir un jeu de paramètres
b) Exécuter l'algorithme pour ***T*** évaluations
c) Examiner le résultat, décider s'il peut être amélioré. Dans l'affirmative aller en a).

Il s'agit bien d'un processus visant à optimiser l'optimiseur au moins pour la fonction ***f***, le problème peut se formuler ainsi :

- Espace de recherche = espace des paramètres
- Fonction à minimiser = fonction qui, en tout point de l'espace de recherche renvoie la performance obtenue après la tentative de minimisation de f.

En théorie, il suffit donc d'utiliser un algorithme pouvant s'appeler lui-même, mais il y a deux écueils. D'une part il faut déjà paramétrer cet algorithme. Un paramétrage avec des valeurs empiriques moyennes peut faire l'affaire. Mais, d'autre part, l'évaluation de chaque point de l'espace de recherche nécessite T évaluations de la fonction f.

7. L'adaptation :

Jusqu'ici nous avons considéré qu'un algorithme d'OEP classique et sans adaptation, c'est-à-dire qui ne modifie pas son comportement selon les informations recueillies lors du processus de recherche. Des essais très nombreux permettent alors de trouver pour chaque fonction du jeu d'essai les paramètres qui donnent le meilleur résultat.

L'excellence qualité de ces résultats montre que les principes de base de l'OEP sont efficaces, cependant, il n'est pas réaliste de penser que l'utilisateur aura toujours la possibilité de chercher longuement un paramétrage adéquat pour son problème, c'est pourquoi les paramétrages optimaux découverts ici sont analysés afin d'en déduire des indices quant aux règles de comportement que devrait suivre un algorithme adaptatif robuste et pratique.

7.1 Pondération à décroissance temporelle :

On rappel les équations de mouvement d'un algorithme d'OEP :

$$\begin{cases} v_d \leftarrow c_1 v_d + c_{max} alea(0,1)(p_d - x_d) + c_{max} alea(0,1)(\mathrm{g}_d - x_d) \\ x_d \leftarrow x_d + v_d \end{cases}$$

Le coefficient c_1 qui représente la confiance que la particule accorde à son mouvement propre, a été toujours considéré constant jusqu'ici, il est donc assez naturel d'essayer de le faire varier, un simple choix au hasard ne répondrait à aucun critère d'adaptation. D'une part cette méthode ne tiendrait pas compte de l'information recueillie et d'autre part définir la distribution de probabilité pour le tirage aléatoire nécessiterait de définir au moins un nouveau paramètre.

Une méthode qui a été très employée est celle consistant à faire diminuer ce coefficient au cours du temps (après chaque itération), typiquement la loi de décroissance donne une valeur tendant asymptotiquement vers zéro, l'idée est que quand le nombre d'itération augmente, alors l'algorithme est probablement en train de converger et donc il vaut mieux faire progresser les particules de plus en plus lentement de façon à ne pas rater l'optimum.

Dans certaines applications cette intuition se relève exacte, mais par rapport à nos exigences nous voyons tout de suite ou le bât blesse : cette méthode repose sur une prédiction auto-réalisatrice ; la convergence va s'améliorer avec le nombre d'itérations, cela est forcement toujours vrai car faire diminuer le coefficient $\boldsymbol{c_1}$ revient globalement à faire diminuer toute les vitesses il va donc bien y avoir une convergence vers un état quasi – stationnaire. Cependant rien ne garantit que l'une des positions obtenues soit le minimum cherché. Et il faut pour chaque problème chercher empiriquement une définition de la fonction de décroissance qui permettra de trouver une solution sans mettre trop de temps mais également sans convergence prématurée.

7.2 Sélection et remplacement :

L'OEP est récente par rapport aux AG, donc on peut en tirer partie de ces derniers par exemple l'analogie en OEP des opérateurs génétiques tels que le croisement et la mutation (vitesse et combinaison d'informations) mais pas la sélection qui est équivalente ici a l'élimination des particules, car en effet l'OEP se base comme nous l'avons montré sur la philosophie de coopération et non pas sur la compétition, néanmoins en recherche le

dogmatisme n'est pas de mise et si une OEP avec sélection se relève intéressante, il ne faut pas hésiter.

La première tentative était en 1998, la sélection consistait à élimine 50% d'individus générés par croisement classique ou par mutation, notons que cette méthode nécessite une taille constante de l'essaim.

Malheureusement, les tests ont montré que les performances ne pouvaient être globalement meilleures que celles de l'OEP classique que si la valeur de ce pourcentage était ajustée pour chaque problème. Il s'agissait donc en définitive d'un paramètre supplémentaire. Cependant cela a permis de montrer que la sélection pouvait en effet parfois améliorer l'OEP.

7.3 Adaptation paramétrique :

On utilise dans cette méthode un coefficient $\boldsymbol{\chi}$ appelé de *constriction* dans les équations de mouvement, c'est-à-dire que les coefficients de confiance sont calculés en fonction d'un seul paramètre $\boldsymbol{\varphi}$

$$\begin{cases} C_1 = \chi = \frac{1}{\varphi - 1 + \sqrt{\varphi^2 - 2\varphi}} \\ C_{max} = \varphi . C_1 \end{cases} \qquad avec\ \varphi > 2$$

Les comparaisons de performances ne se font que localement c'est-à-dire pour chaque particule testée on ne considère que son voisinage (au sens de l'ensemble de ses voisinages), l'idée est que si l'une au moins des particules du voisinage (qui comprend la particule elle même) a « suffisamment » amélioré sa performance, alors on peut supprimer la plus mauvaise particule dudit voisinage. A l'inverse si la meilleure particule n'a pas suffisamment amélioré sa performance, alors on génère une nouvelle particule (totalement au hasard, en l'occurrence).

La taille de l'essaim étant variable, on peut commencer toujours avec le même petit nombre de particules (au moins deux pour que l'adaptation puisse déclencher).

La taille des voisinages est également modifiée, formalisant la règle intuitive voulant que si une particule améliore ses performances elle n'a pas besoin de continuer à s'informer auprès de nombreuses voisines et inversement.

Le paramètre ϕ est ajusté lui aussi après chaque itération, lorsqu'il y a eu amélioration, il est augmenté ce qui diminue les coefficients de confiance et donc restreint le volume explorable par la variable lors du prochain déplacement, et inversement.

L'inconvénient est dans le nombre total des paramètres à la charge de l'utilisateur, il n'y a pas d'amélioration. Cependant il est possible de reprendre ces idées en remplaçant les formules par des règles sans paramètres qualitatives plutôt quantitatives **[36]**.

8. Tribes ou la coopération de tribus :

8.1 Introduction :

Si chaque particule de l'essaim est vue comme le sommet d'un graphe, on peut représenter le lien d'information par un arc de ***B*** vers ***A***. l'arc inverse de ***A*** vers ***B*** peut exister, par ailleurs, toutes les particules ne pointent pas vers A. Nous pouvons ainsi définir des sous ensemble tels que : toute particule pointe (informe) toutes les autres, nous les appellerons ***tribus***. La métaphore étant celle de groupes d'individus de taille variable évoluant dans un environnement inconnu, à la recherche d'un bon emplacement.

8.2 Structure et relations tribales :

Dans TRIBES, l'essaim est divisé en plusieurs sous-essaims appelés tribus. Les tribus sont détaillés différentes, qui évoluent au cours du temps. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Dans le but de prendre une telle décision, les tribus échangent leurs résultats tout au long du traitement. Un tel type de structure est comparable aux structures multi-essaims utilisées dans d'autres algorithmes. Deux types de communications sont donc à définir : la communication intra-tribu et la communication inter-tribu.

Chaque tribu est composée d'un nombre variable de particules. Les relations entre les particules à l'intérieur d'une tribu sont définies par une topologie complètement connectée. Chaque particule connait la meilleure et la plus mauvaise position jamais atteintes par la tribu, c'est-à-dire que chaque particule connait les positions pour lesquelles la tribu a trouvé la plus petite et la plus grande valeur de la fonction objectif. Cette forme de communication est appelée ***communication intra-tribu***.

Au fur et à mesure du traitement, chaque tribu va converger vers un optimum local. Il est donc nécessaire que celles-ci communiquent entre elles pour définir lequel de ces optima est à retenir par l'utilisateur. La communication entre les tribus est établie par l'intermédiaire des meilleures particules de chaque tribu. Ce type de communication est appelé ***communication inter-tribu***.

En résumé, chaque particule est informée par elle-même, par tous les éléments de sa tribu (appelés informateurs internes), et, si cette particule est un chaman (e.g.la meilleure particule d'une tribu), alors elle est aussi informée par les chamans des autres tribus (appelés informateurs externes). Toutes ces positions sont appelées les informateurs de la particule. La mémoire sociale (g) de chaque particule est l'informateur pour lequel la valeur de la fonction objectif est la plus petite.

La Figure 4.3 illustre la topologie du graphe d'information de l'essaim. Les flèches symbolisent la communication inter-tribu alors que les traits matérialisent la communication intra-tribu. Les particules noires sont les chamans de chaque tribu. On verra dans la suite que cette structure est modifiée automatiquement par l'intermédiaire de créations et de destructions de particules **[39]**.

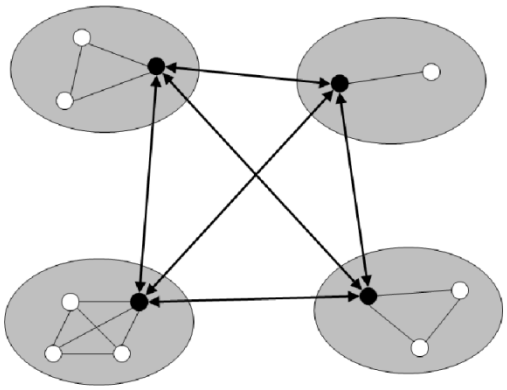


Figure 4.3 : *Réseau d'information. Chaque tribu est un graphe complètement connecté (communication intra-tribu). Les tribus sont reliées par l'intermédiaire de leurs chamans (communication inter-tribu)*

8.3 Évolution des tribus :

Dans le but de définir des règles d'adaptation structurelles pour l'essaim, on définit deux indicateurs de qualité, un pour les particules et un pour les tribus. Une particule est dite bonne si elle a amélioré sa meilleure performance au cours de la dernière itération. Dans le cas contraire, elle est dite neutre. Cet indicateur est purement qualitatif, car il est basé sur l'évolution de la performance et non sur la performance elle-même. En plus de cet indicateur, la meilleure et la plus mauvaise position de la tribu sont stockées.

Une tribu est aussi dite bonne ou mauvaise suivant le nombre de bonnes particules présentes en son sein. Une tribu est dite mauvaise si aucune de ses particules n'a amélioré sa meilleure performance au cours de la dernière itération. Les tribus qui possèdent au moins une bonne particule sont déclarées bonnes avec une probabilité de 0.5, mauvaises sinon.

L'utilisation de ces indicateurs permet de définir les règles de création et de destruction de particules.

8.3.1 Destruction de particules :

Dans un souci de gain de temps, il est important d'évaluer la fonction objectif le moins de fois possible. Notre intérêt est donc de détruire des particules de l'essaim à chaque fois que cela est possible. Ces destructions sont réalisées en espérant que le résultat final ne soit pas altéré. C'est pourquoi seules les plus *mauvaises* particules des *bonnes tribus* sont détruites. Il est ici considéré que de telles particules n'apportent plus d'informations pertinentes à la tribu, et donc à l'essaim, au regard des performances de ses congénères.

Dans le cas d'une tribu composée d'une seule particule, la destruction est réalisée seulement si l'un des informateurs externes a une meilleure performance que la particule considérée. Cela nous assure de ne retenir que les particules qui apportent de bonnes informations. Sur la **Figure 4.4**, la tribu ***T1*** est déclarée *bonne*. La particule ***P***, qui est obligatoirement *la plus mauvaise* de la tribu, est détruite, même si elle en est aussi la meilleure. Cependant, ***P*** ne sera détruite que si son informateur externe $\boldsymbol{M_p}$ possède une meilleure performance. L'hypothèse est que l'information apportée par P est de moins bonne qualité que celle apportée par $\boldsymbol{M_p}$ **[39]**

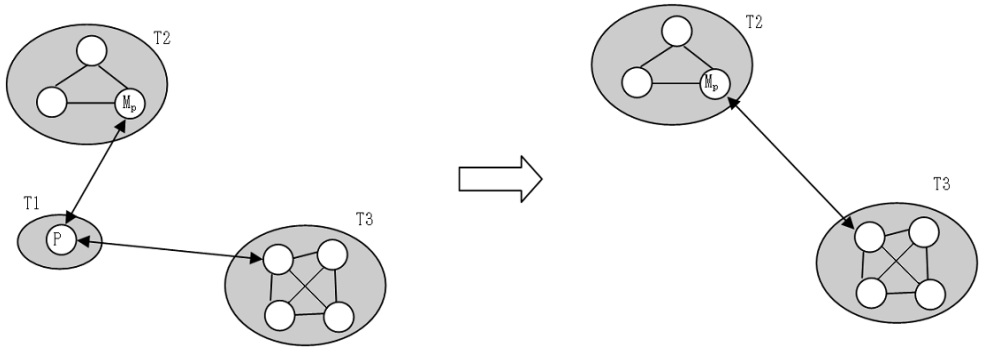


Figure 4.4 : Destruction de particule. La tribu ***T1*** est détruite, les liens d'informations sont redistribués vers ***T2*** et ***T3***

La destruction de particules implique un changement dans le graphe d'information de l'essaim. Toutes les particules informatrices d'une particule supprimée sont redirigées vers la meilleure particule de sa tribu. Dans le cas particulier d'une tribu de taille 1, tous les liens d'informations sont redirigés vers le meilleur informateur de la particule supprimée.

8.3.2 Création de particules :

Le procédé qui permet de créer des particules est assez similaire à celui qui permet d'en détruire. Chaque mauvaise tribu génère des particules. Le nombre de particules générées par chaque mauvaise tribu est défini par l'équation (1). Toutes les particules générées par les différentes mauvaises tribus forment une nouvelle tribu.

$$NB_{particules} = max\left(2, \left|\frac{9.5+0.124\ (D-1)}{tribeNb}\right|\right) \qquad \textbf{(1)}$$

Où ***D*** est la dimension de l'espace de recherche et ***tribeNb*** le nombre de tribus dans l'essaim.

Les mauvaises tribus sont alors reliées à la nouvelle tribu et utilisent la nouvelle information apportée par cette tribu pour améliorer leurs performances. En pratique, le chaman de la nouvelle tribu est un informateur externe des chamans des mauvaises tribus **[39]**.

Deux types de particules sont générés, le type de particule étant sélectionné aléatoirement :

- **Les particules libres :**

Ces particules sont générées aléatoirement à l'aide d'une distribution uniforme soit dans tout l'espace de recherche, soit sur une frontière de celui-ci, ou sur un sommet. Les particules sont générées en utilisant l'équation suivante. Le but est d'apporter de la diversité à la population.

$$\begin{cases} Dans\ tout\ l'espace\ de\ recherche: & X_j = U(x_{min-j}, x_{max-j}), j \in \{1,\dots,D\} \\ Sur\ une\ frontière: & X_j = \begin{cases} U(x_{min-j}, x_{max-j}), & j \in I \subset \{1,\dots,D\} \\ U(\{x_{min-j}, x_{max-j}\}), & j \in J \subset \{1,\dots,D\} \end{cases} \\ Sur\ un\ sommet: & X_j = U(\{xmin-j, xmax-j\}), j \in \{1,\dots,D\} \end{cases}$$

où $U(x_{min-j}, x_{max-j})$ est un réel uniformément choisi dans $[x_{min-j}, x_{max-j}]$ et $U(\{x_{min-j}, x_{max-j}\})$. I et J sont deux sous-espaces qui forment une partition de $\{1, \dots, D\}$. Ces deux espaces sont définis aléatoirement pour chaque particule générée.

- **Les particules confinées :**

Ici, l'idée est presque inverse, il s'agit au contraire d'intensifier la recherche dans une région qui semble intéressante.

Soit x la meilleure particule de la tribu génératrice et $\hat{x}$ sa meilleure position mémorisée. Soit g la meilleure informatrice de x, et $\hat{\mathbf{g}}$ la meilleure position que cette informatrice a mémorisée. Alors la nouvelle particule sera générée au hasard uniformément dans la D-sphère de centre $\hat{\mathbf{g}}$ et de rayon $\|\hat{\mathbf{g}} - \hat{x}\|$.

8.3.3 Fréquence des adaptations :

Les adaptations structurelles ne doivent pas être effectuées à chaque itération. En effet, du temps doit être laissé pour que l'information introduite par la dernière modification de la topologie se propage dans l'essaim. Théoriquement, le temps nécessaire pour que l'information se propage dans tout l'essaim est égal au diamètre du graphe d'information. Cela assure que chaque particule soit directement ou indirectement informée des changements apportés lors de la dernière adaptation.

Cependant, si la taille de l'essaim devient trop importante, ce nombre peut vite devenir très long à calculer. En pratique, si ***NL*** est le nombre de liens d'informations lors de la dernière adaptation, la prochaine adaptation sera effectuée ***NL/2*** itérations plus tard **[39][36]**.

8.3.4 Évolution de l'essaim :

Au début du traitement, l'essaim est composé d'une particule unique, qui constitue une tribu à elle seule. Si, durant la première itération, cette particule n'améliore pas sa performance, de nouvelles particules vont alors être générées pour former une deuxième tribu. ***NL/2*** itérations plus tard, le même procédé est répété. On continue selon ce schéma durant toute l'exécution.

La taille de l'essaim augmente jusqu'à ce que l'essaim trouve des zones « prometteuses » de l'espace de recherche. Plus la taille de l'essaim devient grande, plus le temps séparant deux adaptations va être long. La capacité d'exploration est améliorée et il va être plus facile pour les particules de trouver de bonnes solutions.

Une fois que des zones intéressantes de l'espace de recherche sont trouvées, les plus mauvaises particules vont être progressivement supprimées.

9. Conclusion :

Dans ce chapitre nous avons consacré un intérêt particulier à la méthode d'optimisation par essaim particulaire. Cette jeune méthode, inspirée des déplacements d'animaux en essaims, a rencontré un vif succès depuis sa création. Sa relative simplicité et son efficacité en font un des algorithmes les plus utilisés de nos jours. De nombreux axes de recherches ont pour objet de tirer le meilleur parti du paradigme de l'OEP.

D'abord nous avons commencé par citer son principe ainsi que son origine, et nous avons donné sa première formulation ou bien sa version classique, puis nous avons parlé sur un algorithme itératif adaptatif qui modifie son comportement en fonction de sa découverte progressive, et le programme TRIBES est un exemple de réalisation d'un tel algorithme puisqu' il fonctionne par coopération de tribus.

Chapitre V :

Conception du régulateur flou à l'aide d'une technique d'optimisation

Introduction :

Dans ce chapitre on va exploiter les deux techniques d'optimisation présentées au chapitre **02** et **03** pour la synthèse du contrôleur flou, cela d'une part pour remédier au caractère de subjectivité qui a prévalu lors de la conception du contrôleur précédent et d'autre part pour la recherche des paramètres optimaux du régulateur ainsi obtenir les performances optimales.

Les paramètres du régulateur qui seront objet d'optimisation sont : les conclusions, les centres et les écarts-type des fonctions d'appartenances, les gains de l'erreur, sa variation ainsi que le gain en sortie. Les seuls paramètres fixés sont le nombre de fonctions d'appartenances et leurs formes, ainsi donc la dimension de la table de décision. Si on prend le problème tel qu'il est on aura à optimiser quatre vingt paramètres (49 règles, 7 centres et 7 écart-type pour les deux entrées, 3 gains) ce qui est énorme et non pratique, pour cette raison un codage approprié du problème est indispensable.

I. Codage du problème :

1. Codage de la table de conclusions :

La table des conclusions et une matrice 7×7 sous la forme universelle de Mac Vicar-Whelan, figure suivante :

Δu		e						
		NG	NM	NP	Z	PP	PM	PG
Δe	NG	$-c_1$	$-c_2$	$-c_3$	$-c_4$	$-c_5$	$-c_6$	0
	NM	$-c_2$	$-c_3$	$-c_4$	$-c_5$	$-c_6$	0	c_6
	NP	$-c_3$	$-c_4$	$-c_5$	$-c_6$	0	c_6	c_5
	Z	$-c_4$	$-c_5$	$-c_6$	0	c_6	c_5	c_4
	PP	$-c_5$	$-c_6$	0	c_6	c_5	c_4	c_3
	PM	$-c_6$	0	c_6	c_5	c_4	c_3	c_2
	PG	0	c_6	c_5	c_4	c_3	c_2	c_1

Ainsi nous avons réduit le nombre de conclusions de 49 à 6. De plus on impose aux conclusions l'ordre suivant :

$$c_1 > c_2 > c_3 > c_4 > c_5 > c_6 \quad tel\ que: c_{i=1,\dots,6} \in [0\ ,1]$$

Pour vérifier cette contrainte, nous avons pris :

$$c_k = \sum_{j=k}^{6} c_j$$

Puis on normalise suivant la formule:

$$c_k = \frac{c_k}{\max\ (c_{k=1,\dots,6})}$$

Ainsi on obtient notre table de conclusions normalisée, et symétrique suivant la forme de Mac Vicar-Whelan.

La commande ainsi générée est une commande normalisée et pour l'appliquer sur l'entrée du processus elle doit subir une mise à l'échelle par un gain G_u.

2. Codage des fonctions d'appartenances :

Le nombre de fonction d'appartenance est fixé à sept, de formes gaussiennes symétriques sur des univers de discours normalisés$[-1; 1]$. On suppose aussi que celles de l'extrémité sont centrées autour de $-1\ et\ 1$ respectivement et celle du milieu autour de zéro. Donc ce qui restera à fixer sera les centres des fonctions restantes et la largeur de toutes les fonctions, et cela pour les deux entrées $e\ et\ \Delta e$.

Pour le calcul des centres des fonctions d'appartenance on définit un paramètre d'espacement et pour le calcul de leurs largeurs on définit des paramètres de forme.

2.1 Paramètre d'espacement :

Le paramètre s'espacement PsG définit la manière dont les coordonnées (positions) C_i des points intermédiaires (entre le centre et les extrémités) sont espacées par rapport au centre. Ce paramètre offre une flexibilité d'espacement variable telle que, plus il est supérieur à 1, plus les coordonnées sont davantage rapprochées du centre, et plus il est inférieur à 1, plus les coordonnées sont éloignées du centre. A la valeur 1, les positions sont, donc, réparties de manière uniforme (équidistante) dans l'univers de discours $[-1; 1]$.

Les coordonnées étant, évidemment, de nombre égal à celui des FAs utilisés, nous avons utilisé une formulation de la loi de leur espacement en fonction de la valeur du paramètre PsG **[24].**

D'abord, leurs positions réparties de manière équidistante, notées CE_{qi} , sont exprimées par :

$$CE_{qi} = 2\left(\frac{i-1}{NFAs-1}\right) - 1 \qquad i = 1, \dots NFAs$$

Avec $NFAs$ est le nombre de fonctions d'appartenance, dans notre cas $NFAs = 7$.

Les C_i sont, alors, déterminées en fonction du paramètre d'espacement PsG comme suit:

$$C_i = sign(CE_{qi}) \times |CE_{qi}|^{PsG}$$

Cet effet est illustré par les figures suivantes, pour différentes valeurs du paramètre d'espacement :

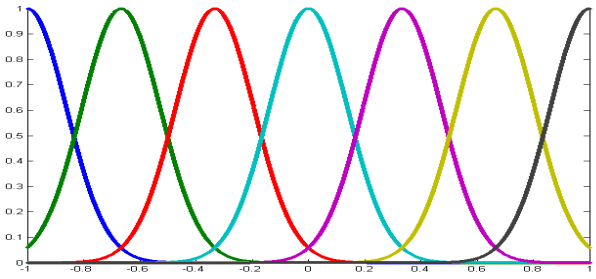


Figure 5.1 FAs avec $PsG = 1$

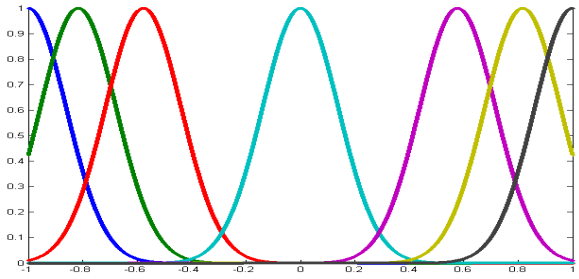


Figure 5.2 FAs avec $PsG = 0.5$

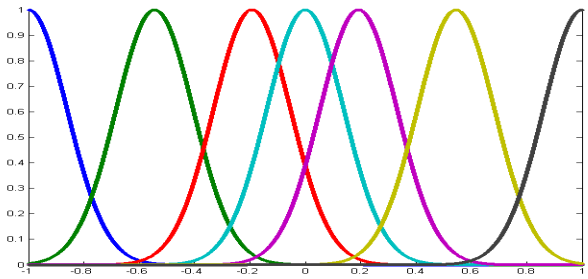


Figure 5.3 FAs avec $PsG = 1.5$

2.2 Paramètres de formes :

Pour la détermination des largeurs des FAs nous avons utilisé un autre paramètre appelé paramètre de forme, à partir duquel les écart-type des gaussiennes peuvent être calculés. Ce paramètre est un nombre réel appartenant à l'intervalle $[0\,;2]$. Pour pouvoir coder toutes les largeurs avec un nombre minimal de paramètres nous avons utilisé une solution qui consiste à effectuer un paramètre de forme, noté PfM , à la FA du milieu et un autre, noté PfE , pour celle de l'extrémité. Les paramètres de forme des FAs intermédiaire, notés PfI , sont calculés à partir de $PfM \;\; et \; PfE$, de sorte à ce qu'ils aient des valeurs intermédiaires à intervalles égaux.

Les paramètres de forme $PfI(i)$ de la ième FA intermédiaire est donné comme suit :

$$PfI(i) = PfM + 2(i-1)\frac{PfE - PfM}{NFA - 1} \quad avec\ i = 1, \dots, \frac{NFA+1}{2}$$

On peut constater que :

$$PfI(1) = PfM \quad et \quad PfI\left(\frac{NFA+1}{2}\right) = PfE$$

Nous constatons sur les figures suivantes, l'effet du paramètre de forme :

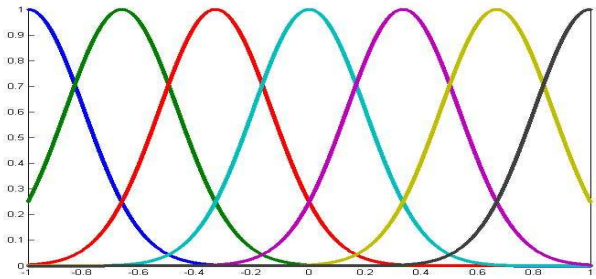


Figure 5.4 FAs avec $PfM = 0.3 \; et \; PfE = 0.3$

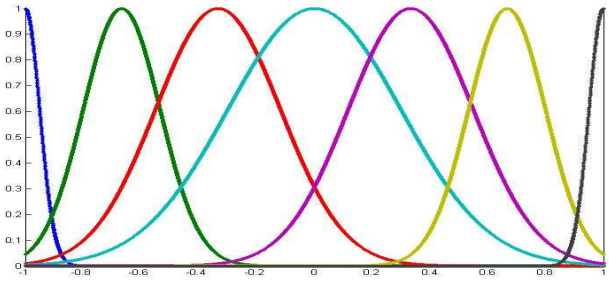


Figure 5.5 FAs avec $PfM = 0.3\ et\ PfE = 0.05$

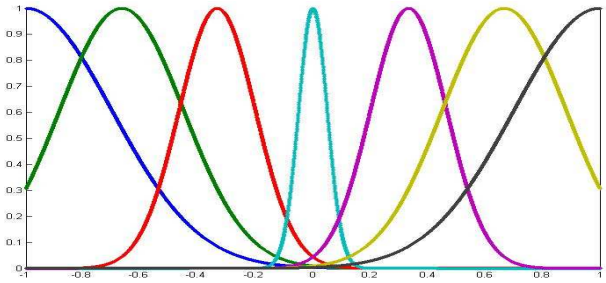


Figure 5.6 FAs avec $PfM = 0.05\ et\ PfE = 0.3$

Ainsi nous aurons codé toutes les FAs avec seulement trois paramètres, et avec ces trois paramètres on obtient des formes très variées de FAs, les figures suivantes montrent qu'avec la combinaison des paramètres d'espacement et de forme on obtient des formes très différentes de FAs :

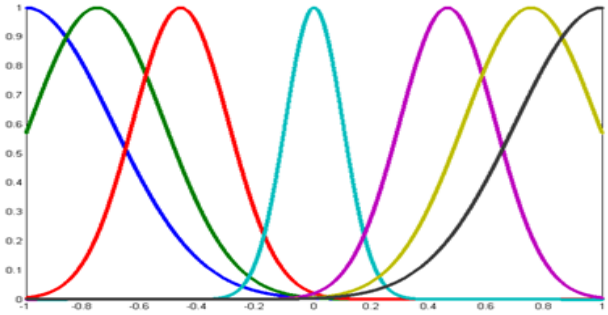


Figure 5.7 FAs avec $PsG = 0.7\ ;\ PfM = 0.1\ et\ PfE = 0.3$

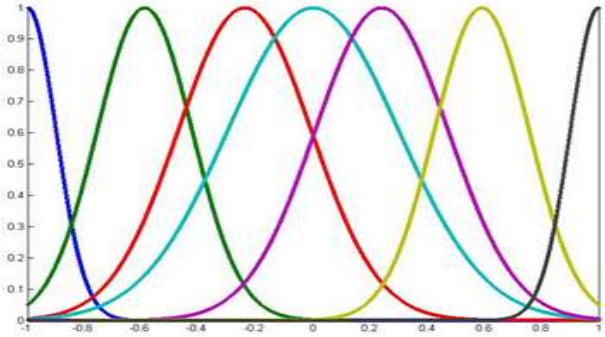


Figure 5.8 FAs avec $PsG = 1.3\ ;\ PfM = 0.3\ et\ PfE = 0.1$

3. Codage des paramètres :

Dans ce qui suit on résumera tout les paramètres à rechercher et on les arrangera dans un vecteur de nombres réels.

Paramètre	Intervalle
$c_1, \dots, c_6$	$[0\,;1]$
PsE	$[0\,;2]$
$PfME \ et \ PfEE$	$[0.01\,;1]$
$Ps\Delta E$	$[0\,;2]$
$PfM\Delta E \ et \ PfE\Delta E$	$[0.01\,;1]$
Gu	$[0\,;15\times10^{-3}]$
Ge	$[0\,;2]$
$G\Delta e$	$[0\,;2]$

Nous aurons enfin chaque individu ou particule sous la forme d'un vecteur de dimension égale à 15.

c_1	c_2	c_3	c_4	c_5	c_6	PsE	$PfME$	$PfEE$	$Ps\Delta E$	$PfM\Delta E$	$PfE\Delta E$	Gu	Ge	$G\Delta e$

Cette codification nécessite une recherche des contrôleurs optimaux dans un espace vaste de dimension 15, ce qui est un véritable défi pour l'algorithme d'optimisation.

4. Fonction objectif :

L'évaluation de la performance d'un CF donné à l'issue de chaque itération de l'exécution de l'AG (ou PSO) est basée sur le résultat (réponse) de la simulation de la commande en concentration du gaz en sortie de la colonne.

Les valeurs des erreurs par rapport à la référence pour chaque instant d'échantillonnage, ainsi qu'éventuellement les instants de leurs prélèvements serviront au calcul de la fitness de l'individu (particule).

Parmi les index de performance à usage courant en automatique, exprimés en termes de minimisation, nous pouvons citer des critères tels que IAE (intégrale des erreurs absolues), SE (intégrale des erreurs carrées) et ITAE (intégrale des erreurs absolues multipliées par le temps) ; chacun de ces critères possédant ses propres performances.

Nous avons utilisé le critère $ITAE$ qui est, d'ailleurs, utilisé dans la plupart des travaux, concernant l'optimisation des CFs, rencontrés dans la littérature. Nous sommes même allés à utiliser le critère IT^2AE pour répondre à certaines spécificités de notre système, les deux critères sont formulés comme suit :

$$ITAE = \int_{t_0}^{t_f} t \times |e(t)|\, dt = \int_{t_0}^{t_f} t \times |y_r(t) - y(t)|\, dt$$

$$IT^2AE = \int_{t_0}^{t_f} t^2 \times |e(t)|\, dt = \int_{t_0}^{t_f} t^2 \times |y_r(t) - y(t)|\, dt$$

Ou en discret :

$$ITAE = \sum_{k=0}^{k=N} kT_e \times |e(kT_e)|$$

$$IT^2AE = \sum_{k=0}^{k=N} (kT_e)^2 \times |e(kT_e)|$$

Avec T_e est le temps d'échantillonnage, et $N \times T_e = t_f$.

Nous avons eu recours au critère IT^2AE afin de réduire les oscillations qui persistent autour de la référence dans la bande des 5%.

La fonction implémentant la fitness (fonction objectif) du CF appliquée à notre système comprend 4 étapes :

- Décodage des individus (particules) et calcul des paramètres du CF (centres, écart-types, et des conclusions).
- Génération du CF à partir des paramètres trouvés.
- Simulation du système en boucle fermée commandé par le CF trouvé.
- Calcul de la performance du CF à partir des vecteurs des erreurs par rapport à la référence et des instants de leurs prélèvements.

Il est important de signaler que le temps d'exécution que prennent les simulations est énorme, et par conséquent nous étions obligés de travailler avec des populations (essaims) de taille réduite et de réduire aussi le nombre de générations (itérations) à une dizaine. Dans le cas où on a un comportement indésirable, des oscillations, on pénalise cette réponse en augmentant le poids de l'erreur dans la fonction objectif.

A noter que le temps d'échantillonnage a été pris $T_e = 0.5\ secondes$ et la durée de simulation des réponses $t_f = 30\ secondes$.

II. Application de l'algorithme génétique :

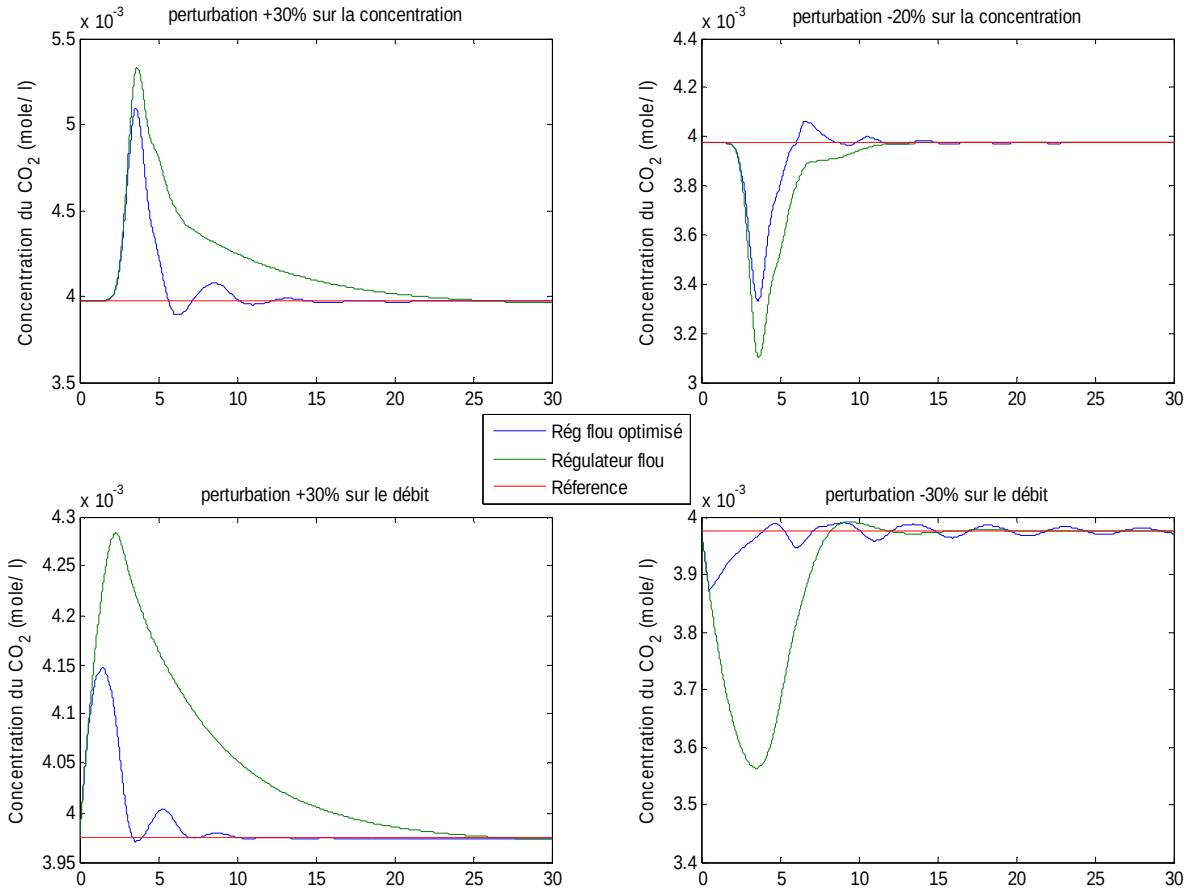
Nous avons développé un algorithme génétique afin de résoudre le problème d'optimisation qui consiste en la recherche des paramètres optimaux du contrôleur flou.

Nous avons choisi le paramétrage suivant pour l'algorithme génétique :

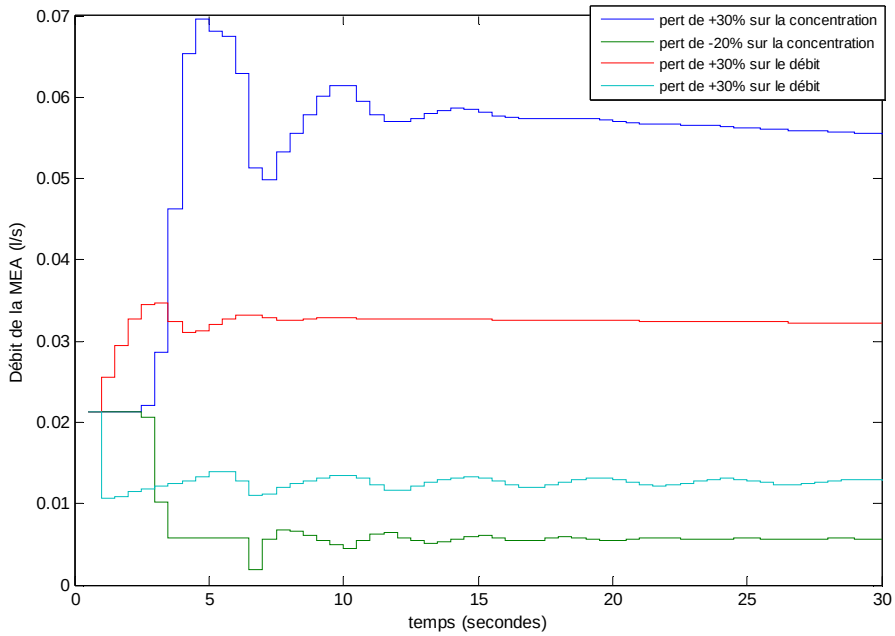
1) **Taille de la population :** elle va de 10 à 25 individus, cette taille semble un peu petite pour un algorithme génétique mais vu le temps énorme de simulation on ne peut l'augmenter beaucoup.
2) **Opérateur de croisement et de mutation :** le croisement considéré est le croisement uniforme barycentrique avec une probabilité allant de 0.7 à 0.8. l'opérateur de mutation est mixé avec celui de croisement et par conséquent la mutation ne se fait que lors du croisement avec une probabilité d'environ 0.1 (c'est-à-dire, il y a que les enfants de la génération courante qui subissent une mutation possible).
3) **Sélection :** la sélection se fait par la roulette biaisée avec la technique du *scaling* exponentiel (relations 3.1 dans le chapitre 03). En effet cette technique s'est avérée très efficace car c'est une sélection adaptative dépendant de l'indice de la génération courante, ainsi pour les premières générations aucun individu n'est vraiment favorisé ce qui renforce le caractère exploration de l'espace de recherche et puis vers la fin, dernières générations, on favorise au contraire beaucoup les bons individus ce qui augmente la vitesse de convergence.
4) **Critère d'arrêt :** comme il n'y a pas de critère d'arrêt universel assurant la convergence de l'algorithme on a choisi de superviser l'algorithme et de décider de son arrêt dès qu'il y aura stationnarité pour plusieurs générations, mais dans tout les cas le nombre maximum de génération ne dépasse pas 30.

Les figures suivantes, montrent les réponses des régulateurs optimaux trouvés pour différentes perturbations en débit et en concentration du gaz en bas de la colonne. On représente aussi sur les mêmes figures les commandes générées et on remarque qu'elles sont admissibles.

- 01 :

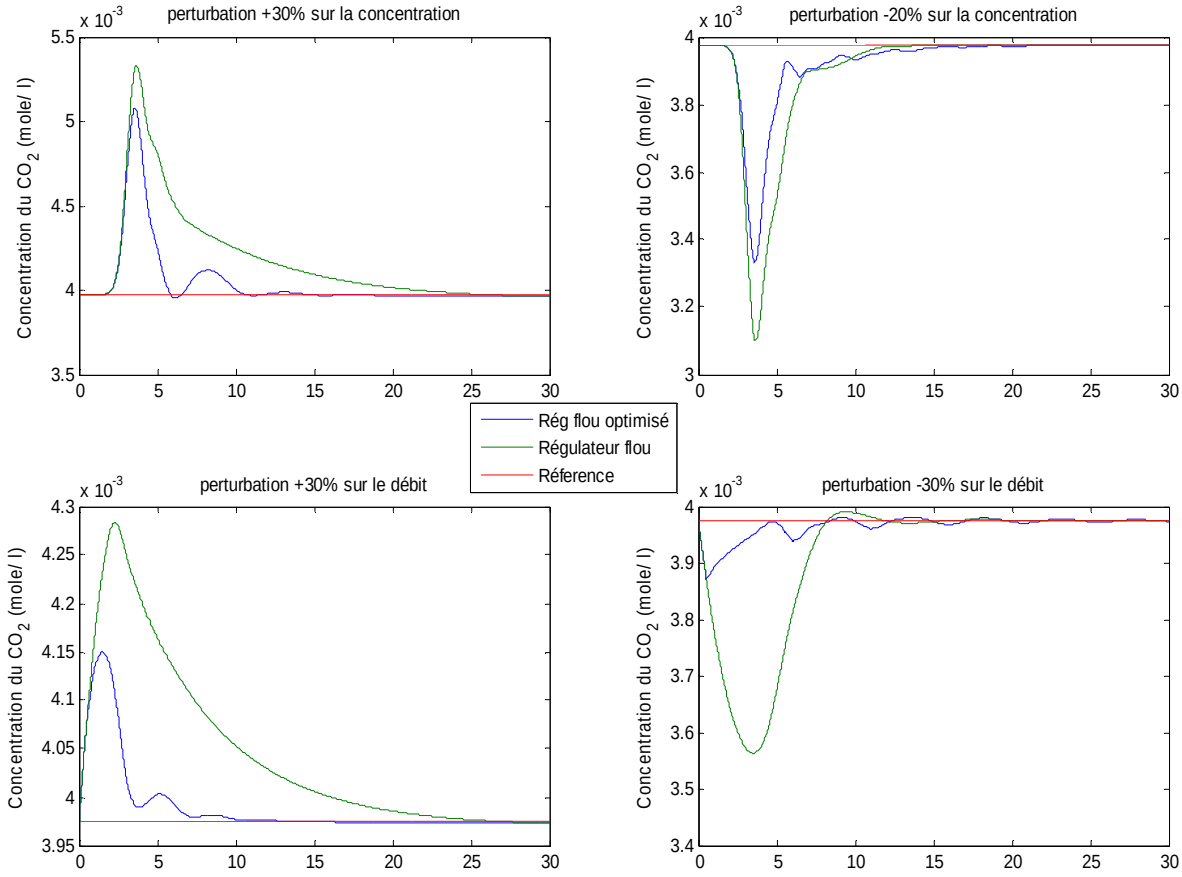


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

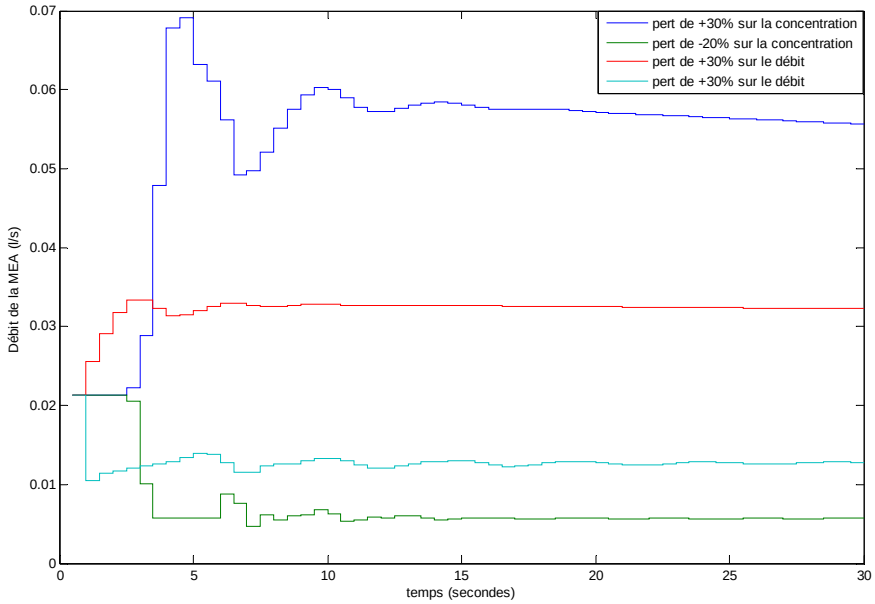


Commandes générées par le régulateur (débit de la MEA).

- 02 :

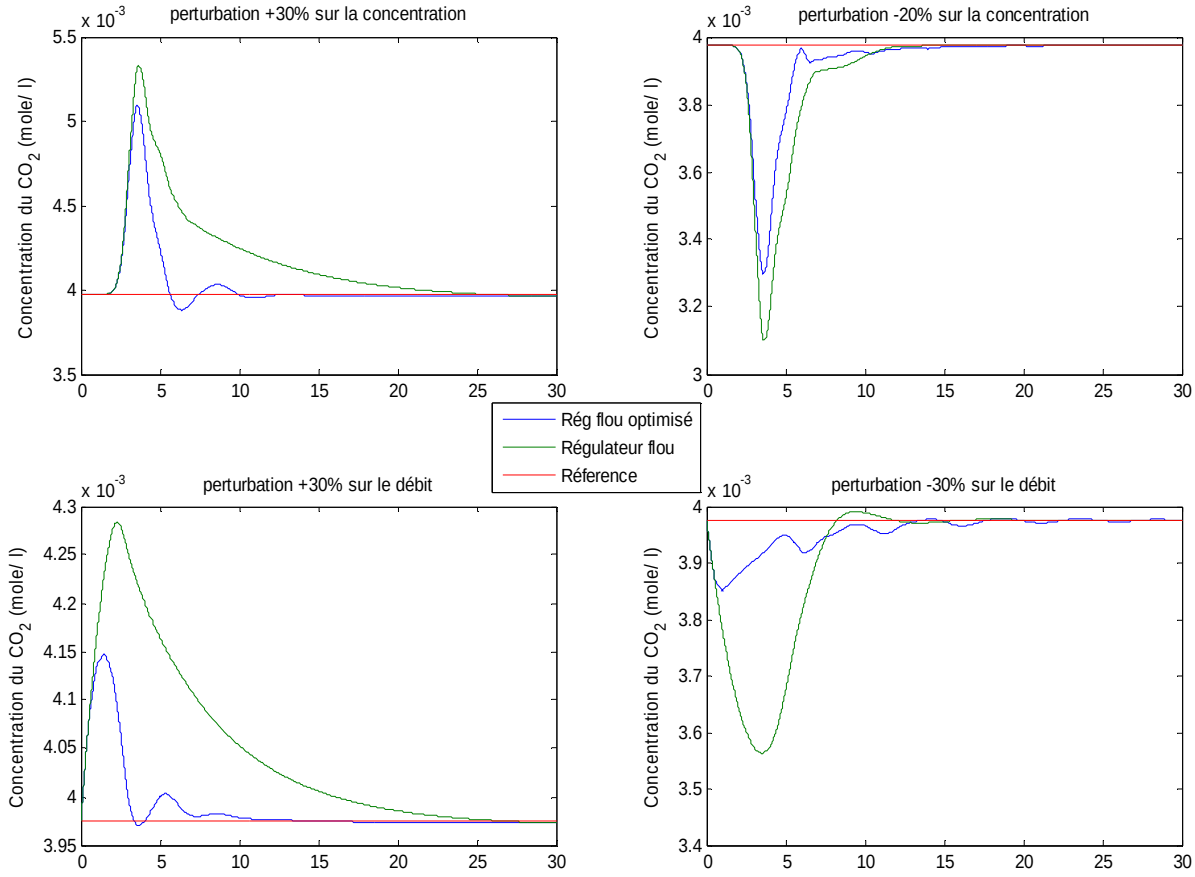


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

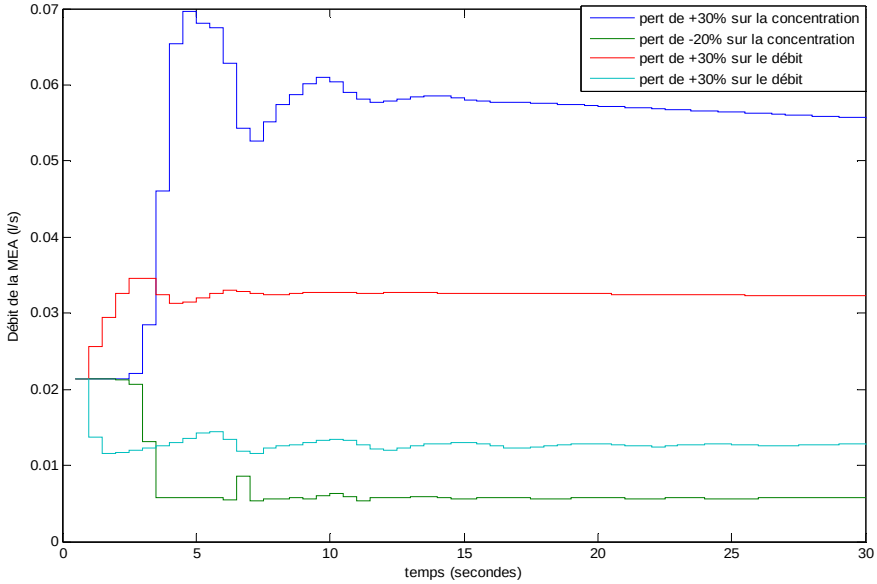


Commandes générées par le régulateur (débit de la MEA).

- 03 :

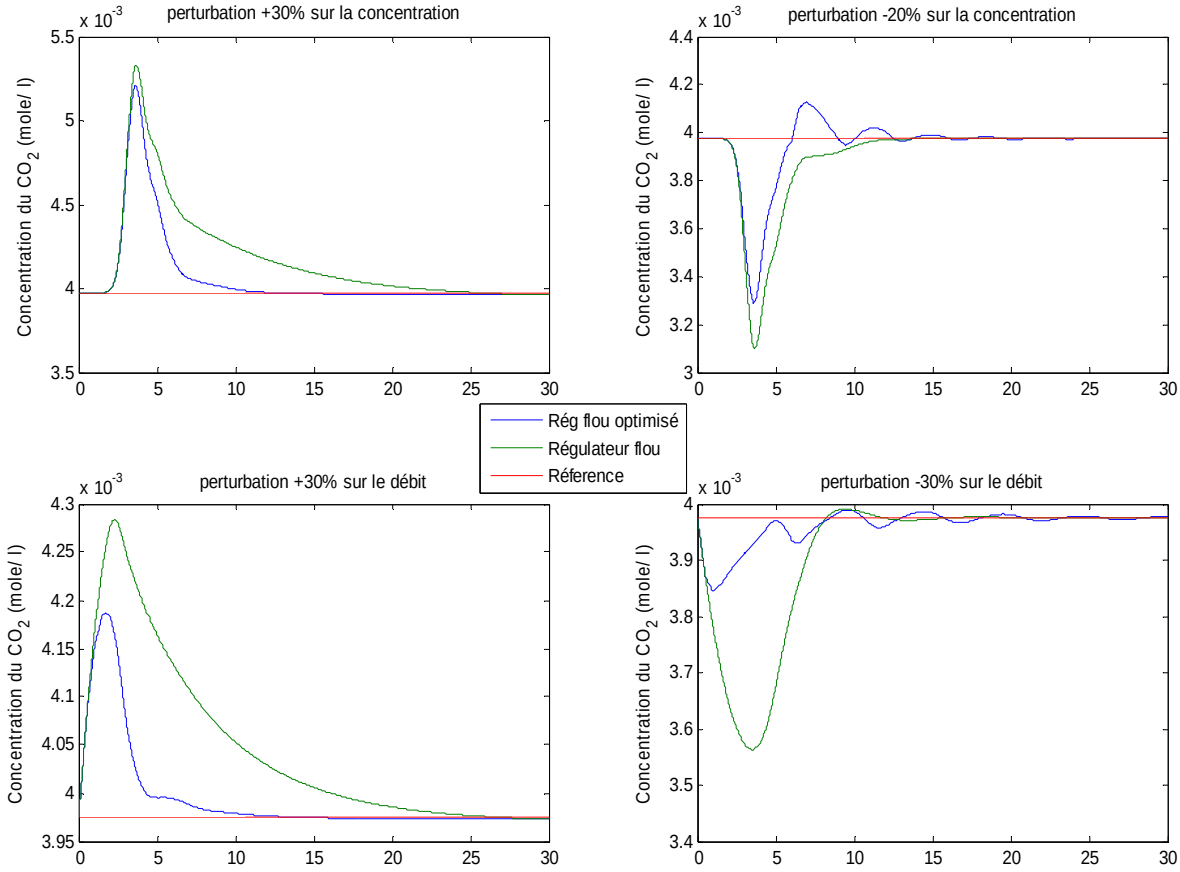


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

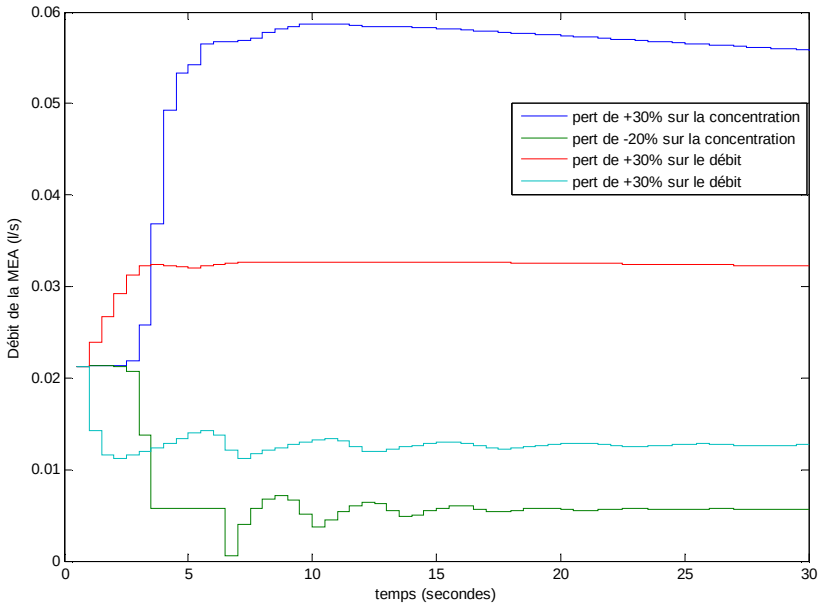


Commandes générées par le régulateur (débit de la MEA).

- 04 :



Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.



Commandes générées par le régulateur (débit de la MEA).

Les figures précédentes montrent que les différents jeux de paramètres optimaux auxquels aboutis l'algorithme donnent des régulateurs flous de dynamiques très proches, ce qui renforce l'hypothèse de la convergence de notre algorithme.

On remarque bien sur les figures que les contrôleurs optimisés comparés au régulateur flou « standard», donné dans le deuxième chapitre, améliorent énormément les performances en atténuant le premier dépassement et en réduisant considérablement les temps de réponse. Les commandes sollicitées restent toujours admissibles.

D'après ces différents constats, on peut déjà conclure sur l'efficacité et l'aboutissement de l'application des algorithmes génétiques pour la synthèse d'un contrôleur flou optimal.

III. Application de l'algorithme PSO :

Nous avons développé une version de l'algorithme PSO dédiée à la tache de la recherche d'un contrôleur flou optimal pour la commande de la colonne d'absorption, on s'est restreint à la version de base de PSO vu qu'elle répond déjà assez bien aux exigences de notre problème.

Les paramètres de l'algorithme sont choisis les suivants :

1) Coefficients de confiance :

$$\begin{cases} C_1 = \chi = \dfrac{1}{\varphi - 1 + \sqrt{\varphi^2 - 2\varphi}} \\ C_{max} = \varphi . C_1 \end{cases}$$

Avec $\varphi = 2.11$, $C_1 = 0.6282 \quad et \;\; C_{max} = 1.3256$

Ce choix est tiré de la littérature et il y a une forte théorie derrière **[36]**.

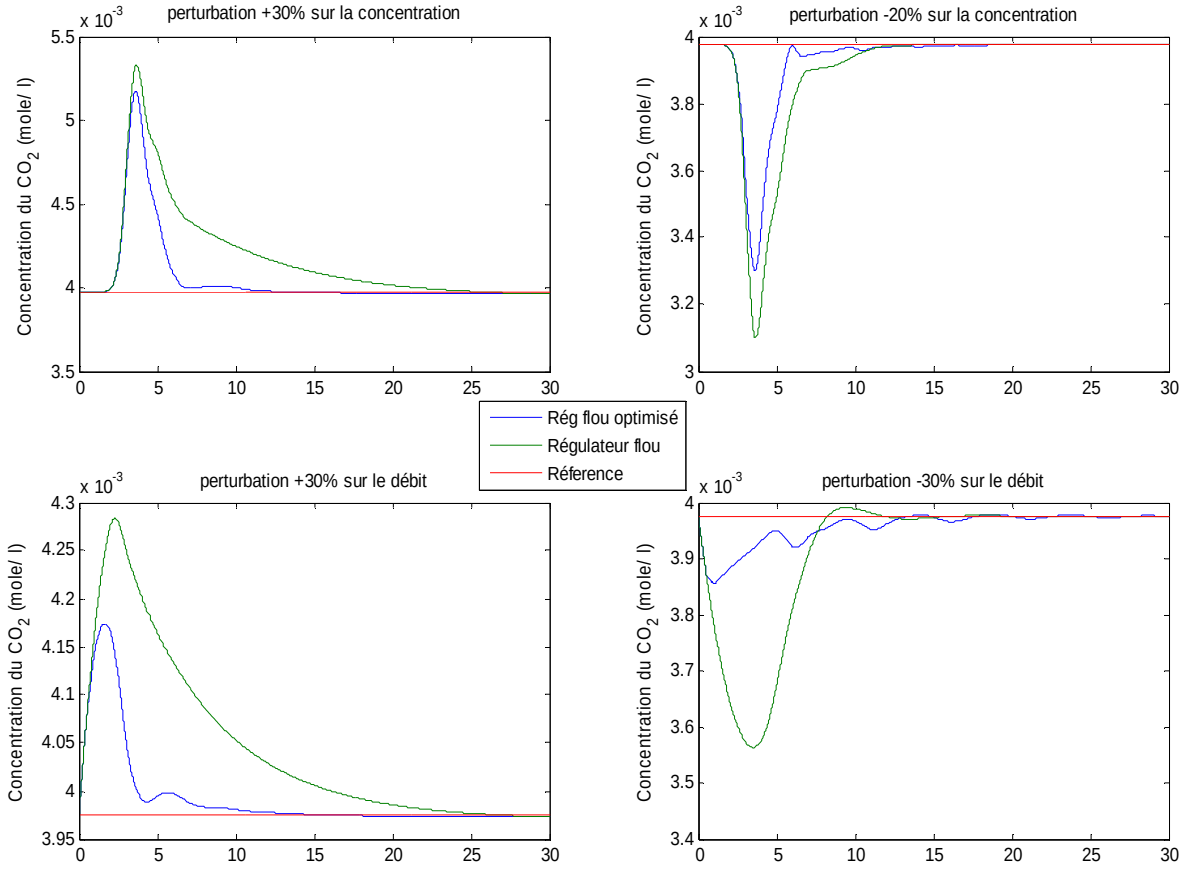
2) Taille de l'essaim : contrairement aux algorithmes génétiques, PSO ne requiert pas une taille d'essaims importante par conséquent le choix d'une taille de 10 à 15 a suffit.

3) Critère d'arrêt : comme il n'y a pas de critère d'arrêt universel garantissant la convergence de l'algorithme, nous avons fixé le nombre maximum d'itération à 30, tout en gardant la possibilité de l'interrompre si un comportement indésirable survient (stationnarité).

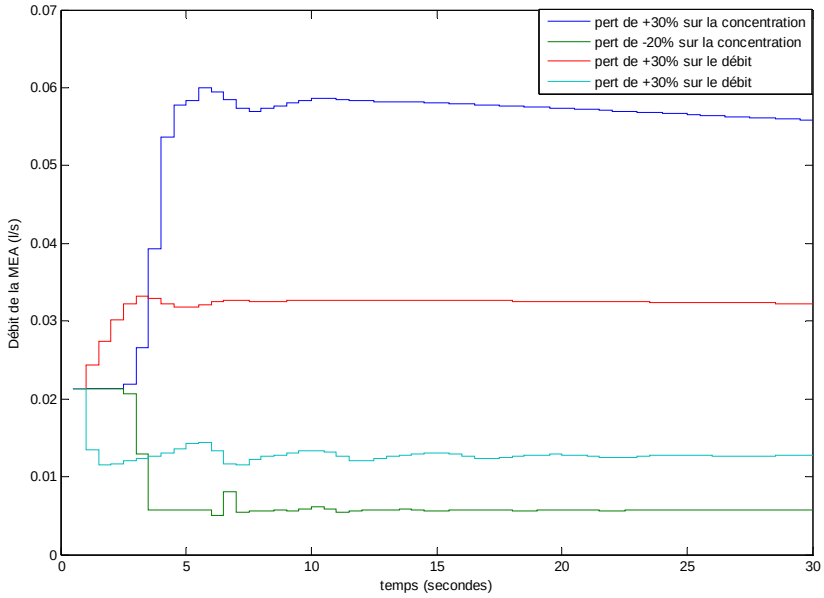
A ce stade déjà on remarque que l'algorithme PSO possède moins de paramètres à fixer et requiert une taille de population moins importante, c'est-à-dire un nombre d'évaluation de la fonction objectif plus petit et par conséquent un temps d'exécution moins important.

Les figures suivantes montrent le comportement de différents contrôleurs optimisés en boucle fermée en réponse à différentes perturbations sur la concentration et le débit du gaz en entrée de la colonne.

- 01 :

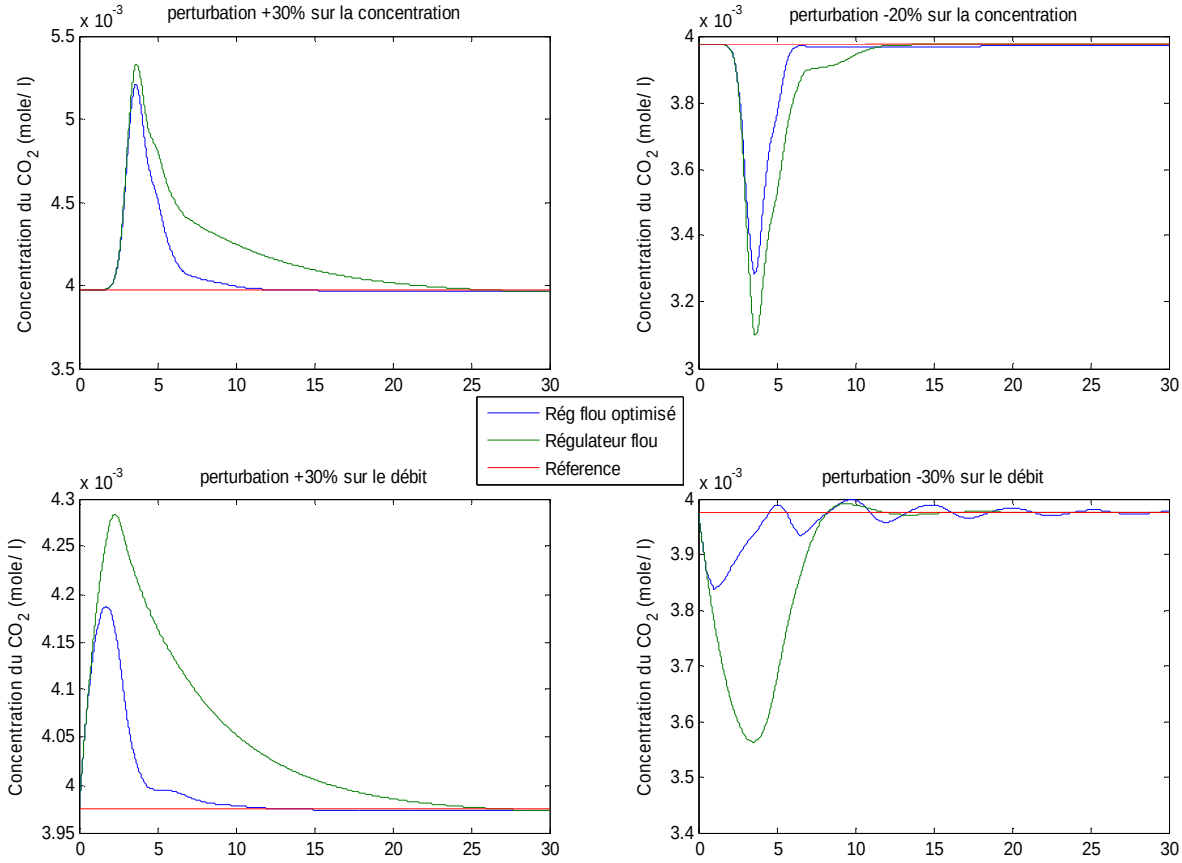


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

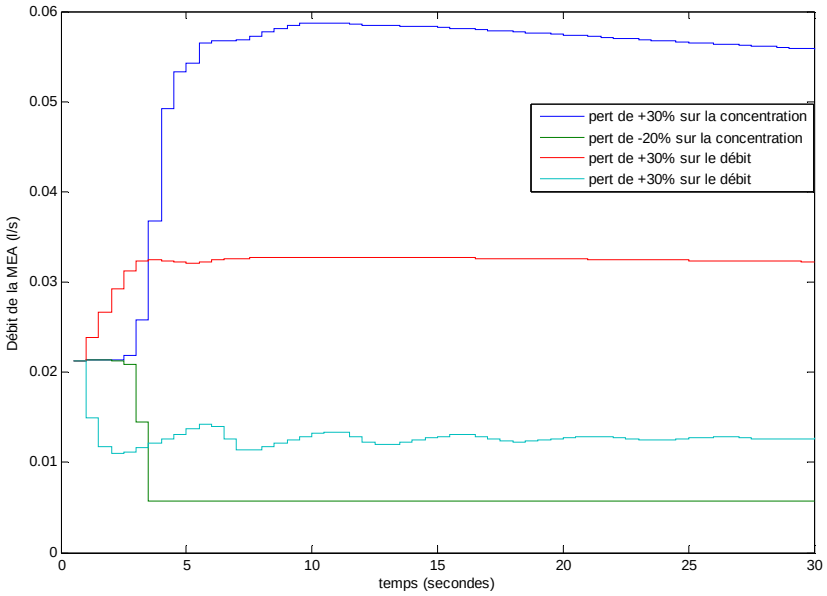


Commandes générées par le régulateur (débit de la MEA).

- 02 :

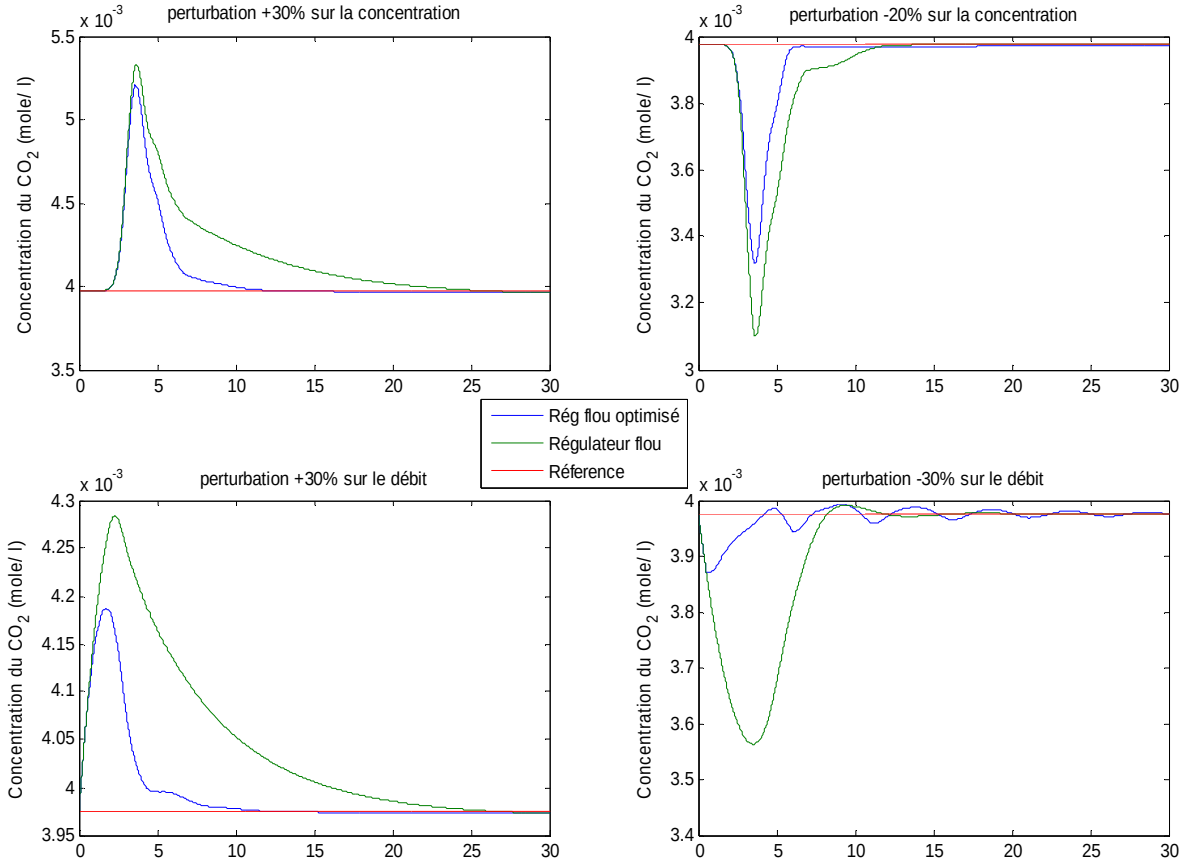


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

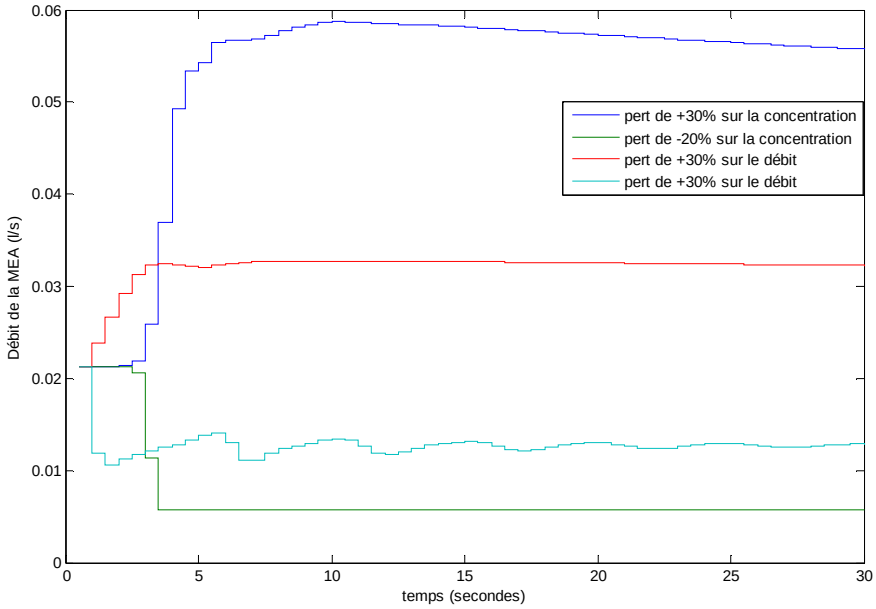


Commandes générées par le régulateur (débit de la MEA).

- 03 :

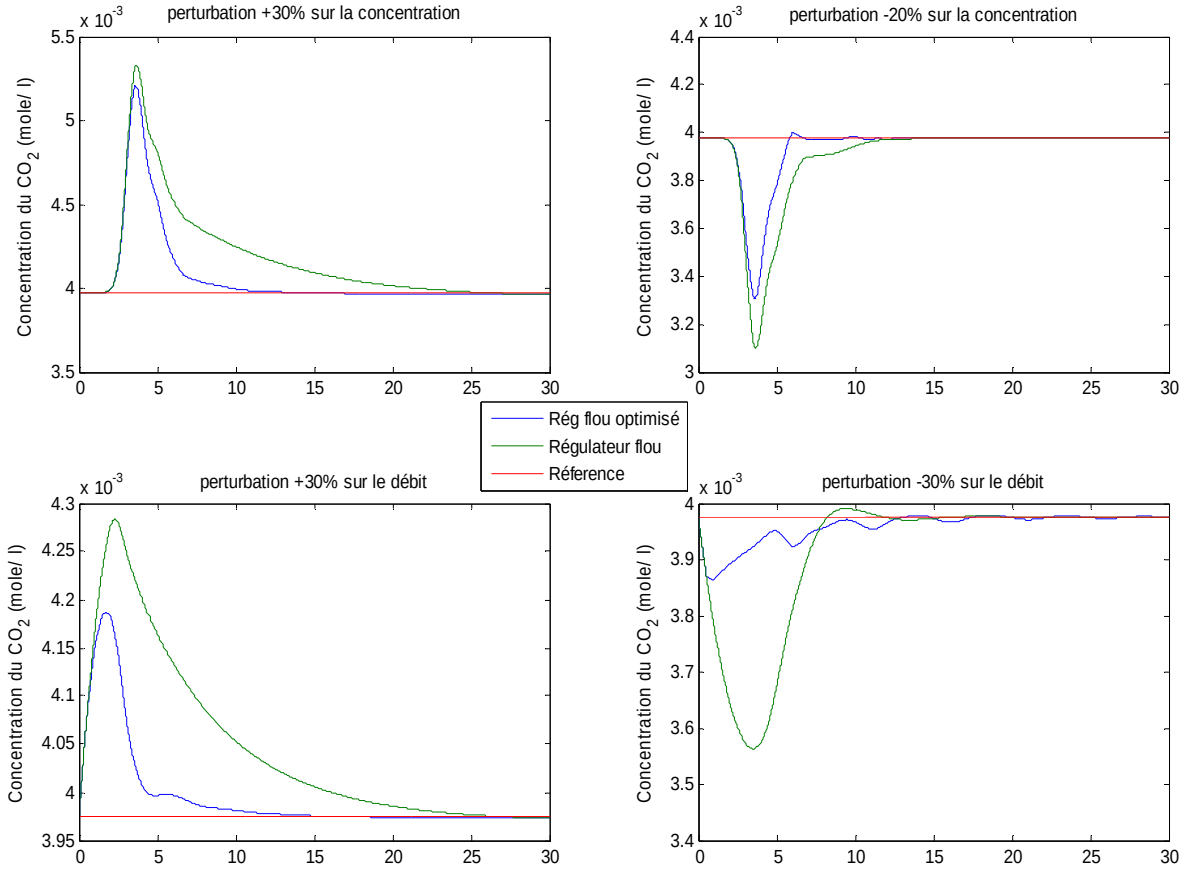


Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.

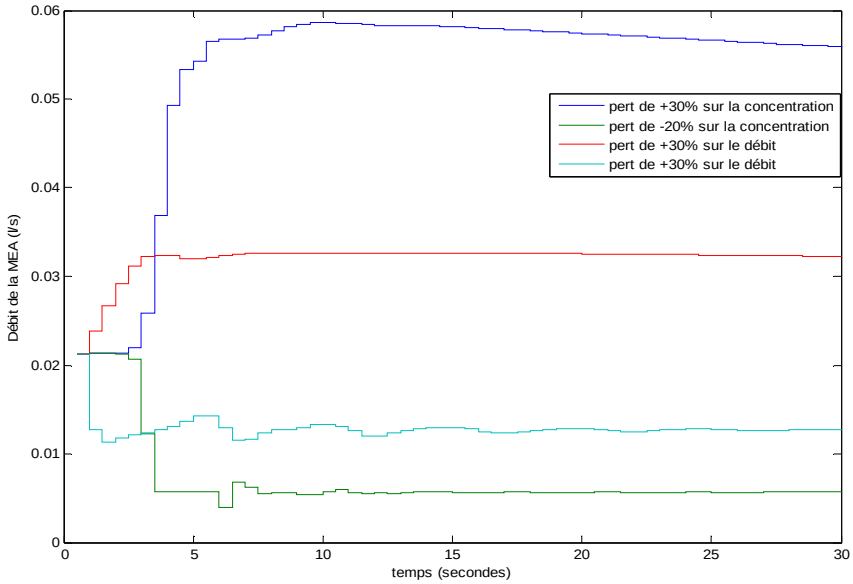


Commandes générées par le régulateur (débit de la MEA).

- 04



Réponses à différentes perturbations sur la concentration et le débit du gaz en entrée.



Commandes générées par le régulateur (débit de la MEA).

Les différentes réponses montrent bien que les régulateurs flous optimisés obtenus améliorent les performances en boucle fermée, et rejettent les perturbations rapidement tout en atténuant le premier dépassement. Les commandes appliquées sont admissibles et relativement lisses par rapport à celles générées par les régulateurs flous optimisés par l'algorithme génétique.

Les régulateurs flous trouvés sont de dynamiques très proches ce qui nous permet de conclure sur la convergence de l'algorithme bien qu'en travaillent souvent avec des tailles d'essaims réduite (10 particules) et un nombre d'itération petit (10 à 15).

IV. En guise de comparaison :

L'objet de ce paragraphe est d'essayer de faire une petite comparaison entre les algorithmes génétiques et l'algorithme *PSO*, bien que ceci n'est pas aussi systématique. En effet pour comparer deux algorithmes d'optimisation, on doit performer de nombreux tests et sur des problèmes de natures très différentes, puisque les performances de ces algorithmes sont *problem dependent*. Alors un algorithme qui est très efficace pour un tel problème peut donner des résultats médiocres pour un autre problème et vice versa.

L'optimisation par essaims particulaires *PSO* est une technique d'optimisation très jeune et prometteuse et il reste encore plusieurs voix à explorer, à savoir la topologie de l'essaim, l'essaim mémoire et d'autres variantes en relation étroite avec l'intelligence artificielle. Alors comparée aux algorithmes génétiques qui sont plus ou moins vieux, on pense que la méthode *PSO* ouvre de nouveaux horizons et de nouvelles perspectives.

Dans notre travail, nous avons remarqué que l'algorithme PSO semble plus efficace. En effet il requiert une taille d'essaim plus petite que la taille de population et un nombre d'itérations réduit par rapport aux nombre de générations pour l'algorithme génétique, ceci implique un temps de calcul nettement réduit.

Sur le plan complexité, l'algorithme *PSO* est plus simple, du fait qu'il n'y a pas beaucoup de paramètres à fixer contrairement aux algorithmes génétiques qui demandent de trouver les bons opérateurs génétiques, un bon opérateur sélection et bien évidement les probabilités de l'application des différentes opérations.

D'un autre point de vue, l'approche *PSO* et génétique ne sont pas forcément en concurrence et elles peuvent bien coopérer. On trouve déjà actuellement quelques versions allant dans ce sens, des versions de *PSO* avec sélection existent.

En se basant sur une comparaison des différentes réponses portées ci-dessus, nous avons choisi que notre régulateur flou optimal soit le suivant (la $4^{ème}$ réponse de la série PSO figure) :

- **Forme des fonctions d'appartenance :**

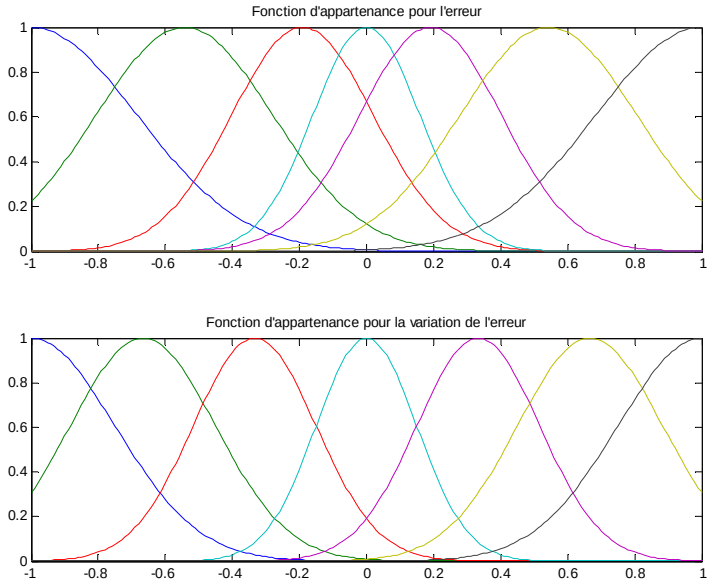


Figure 5.9 Forme des FAs pour l'erreur et sa variation.

- **Table des conclusions :**

Δu		e						
		NG	NM	NP	Z	PP	PM	PG
Δe	NG	-3.000	-2.1908	-1.8006	-1.4579	-1.1856	-0.3048	0
	NM	-2.1908	-1.8006	-1.4579	-1.1856	-0.3048	0	$+0.3048$
	NP	-1.8006	-1.4579	-1.1856	-0.3048	0	$+0.3048$	$+1.1856$
	Z	-1.4579	-1.1856	-0.3048	0	$+0.3048$	$+1.1856$	$+1.4579$
	PP	-1.1856	-0.3048	0	$+0.3048$	$+1.1856$	$+1.4579$	$+1.8006$
	PM	-0.3048	0	$+0.3048$	$+1.1856$	$+1.4579$	$+1.8006$	$+2.1908$
	PG	0	$+0.3048$	$+1.1856$	$+1.4579$	$+1.8006$	$+2.1908$	$+3.000$

- **Les gains :**

Le gain de la commande : $Gu = 1.2 \times 10^{-3}$.

Le gain de l'erreur : $Ge = 1.0$.

Le gain de la variation de l'erreur : $G\Delta e = 2.0$.

- **La caractéristique du régulateur :**

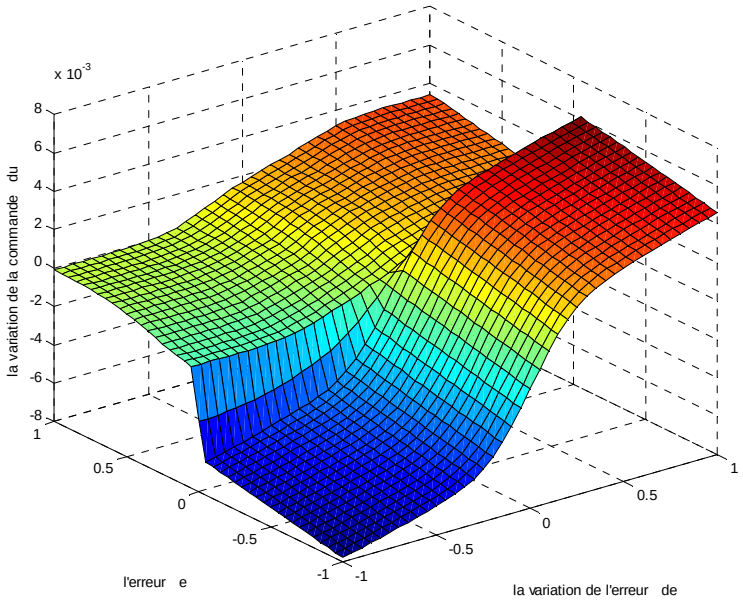


Figure 5.10 caractéristique non linéaire du régulateur flou optimisé.

On remarque bien que la caractéristique du régulateur est non linéaire, cela est un résultat attendu, mais on remarque aussi qu'elle présente une forte asymétrie par rapport au signe de l'erreur, ceci semble logique du moment où la dynamique du processus est aussi asymétrique.

V. Conclusion :

Dans ce chapitre nous avons illustré les résultats de l'application des deux algorithmes d'optimisation pour la synthèse d'un régulateur flou.

Les régulateurs flous optimisés obtenus présentent des performances en boucle fermée nettement améliorées par rapport au régulateur flou de départ. En effet, ils rejettent les perturbations plus rapidement et tolèrent un premier dépassement plus petit. Pour les perturbations sur la concentration de gaz en entrée l'amélioration et surtout sur le temps de réponse, alors nous sommes arrivé à le réduire d'environ $45\ \%$. Pour les perturbations sur le débit, l'amélioration est plus visible. Pour les perturbations positives sur le débit le temps de réponse a été réduit de 40% et le premier dépassement de 30%, pour les perturbations négatives le premier dépassement a été réduit de 70% au prix du même temps de réponse.

Nous pouvons conclure que la synthèse d'un contrôleur flou par une technique d'optimisation est une bonne solution, sachant que la colonne d'absorption est un système complexe régit par des équations aux dérivées partielles de type hyperbolique fortement non linéaire, une telle approche pour sa commande semble très efficace et les résultats obtenus sont très satisfaisants.

Conclusion générale

L'objectif de notre travail a été la mise en œuvre de deux techniques d'optimisation de la commande floue, l'une est la méthode d'optimisation par algorithmes génétiques, et l'autre est celle de l'optimisation par essaims particulaires, plus récente que la première et actuellement fait l'objet de nombreux travaux de recherche. Nous nous sommes penchés sur le procédé d'absorption du CO2 par la monoéthanolamine qui est un procédé de séparation (épuration) largement utilisé dans l'industrie de nos jours.

La commande la plus répandue dans l'industrie pour le pilotage des colonnes d'absorption est la commande classique PI, bien que cette commande ait montrée son efficacité elle est à remplacer dès que des performances plus élevées sont désirées. Cela exige la recherche de lois de commande plus sophistiquées telles que la commande par logique floue.

A cette fin, nous avons commencé par développer un modèle mathématique de la colonne d'absorption en se basant sur les lois fondamentales de la chimie-physique du processus. Le modèle obtenu est régi par un système d'équations aux dérivées partielles de type hyperbolique fortement non linéaires.

Par la suite, et après un bref rappel sur la synthèse des régulateurs flous, nous avons appliqué une commande floue sur notre système, et nous avons remarqué qu'elle est un peu plus performante par rapport à la commande classique PI, ceci après des simulations pour le rejet de perturbations en boucle fermée.

Dans le souci d'améliorer d'avantage les performances, nous avons eu recours à deux techniques d'optimisation : l'optimisation par algorithmes génétiques et l'optimisation par essaims particulaires (*particle swarm optimization*).

Dans le chapitre 03 nous avons donnés les bases des algorithmes génétiques ainsi que les différentes étapes le constituant. Par la suite dans le chapitre suivant les bases et le principe de l'optimisation par essaims particulaires sont donnés, ainsi que quelques variantes de cette technique qui est plus récente que les algorithmes génétiques. Notre objectif dans l'application de ces deux techniques d'optimisation est double, premièrement faire connaitre l'optimisation par essaims particulaire qui est plus récente et moins connues que les algorithmes génétiques et monter son efficacité. Deuxièmement essayer de faire une comparaison entre les deux techniques.

Dans le dernier chapitre, les deux techniques ont été appliquées pour la recherche des paramètres optimaux du contrôleur flou à savoir la table des conclusions, la forme et l'espacement des fonctions d'appartenances et les différents gains. Les deux techniques ont été appliquées avec succès et nous avons pu obtenir des contrôleurs flous nettement plus

performants que les régulateurs de départ ce en réponse à différentes perturbations sur la concentration et le débit du gaz en entrée de la colonne.

Bien que les résultats obtenus dans ce travail soient très satisfaisants et prometteurs, il gagnerait énormément à être enrichi par d'éventuelles extensions, telles que :

- L'extension de la modélisation à des colonnes industrielles et l'application des commandes synthétisées à des applications réelles.
- Développement d'outils de simulation plus rapides des équations aux dérivées partielles afin de pouvoir appliquer les algorithmes d'optimisation avec plus d'efficacité.
- Développement d'une version hybride AG-PSO qui bénéficiera des avantages des deux techniques.
- Développement d'une version adaptative pour l'algorithme d'optimisation par essaims particulaires (PSO).

Bibliographie

[1] R.HADDOUCHE : « simulation et contrôle d'une colonne d'absorption ». Mémoire de Magister. ENP juillet 2006.

[2] S.BEZZAOUCHA : « différentes stratégies de commande floue appliquées a une colonne d'absorption ». Mémoire de Magister. ENP 2007.

[3] Y.KELLOU, M.BELFEGAS : « commande et identification de la colonne d'absorption par réseaux de neurones artificiels ». PFE ENP juin 2007.

[4]A.ILTEN, A.BELAIDI : « Commande par logique floue et Neuro floue d'une colonne d'absorption». PFE ENP juin 2006.

[5] R.ILLOUL, A.SELATNIA, S. ABERKANE, **D.BABA AMI :** « SIMULATION DE LA COMMANDE PAR LOGIQUE FLOUE ET NEURO-FLOUE D'UNE COLONNE D'ABSORPTION REACTIVE ». Rapport de recherche LCP. ENP.

[6] R.ILLOUL, A.SELATNIA, S.BEZZAOUCHA : « Regulation of an Absorption Packed Column of CO_2 using Discrete Fuzzy Input-Output Linearization». Rapport de recherche LCP. ENP.

[7] S.SEDDARI: « simulation d'une colonne à garnissage d'absorption du CO_2 par une solution aqueuse de monoéthanolamine à 25° et 40°c ». Mémoire de Magister. ENP 2004.

[8] W.L.LUYBEN:«chemical reactor design and control».Edition John Wiley & Sons 2007

[9] Help Matlab 7.1®.

[10] P. BORNE, J.ROZINOER, J-Y.DIEULOT, L.DOBOIS : « Introduction à la commande floue ». Edition TECHNIP 1998.

[11] J. GODJEVAC : « Idées nettes sur la logique floue ». Presse polytechnique et universitaires romandes 1999.

[12] R. LONGCHAMP : « commande numérique des systèmes dynamiques ». Presses polytechnique et universitaires Romandes. Lausanne 2006.

[13] BÜHLER : « Réglage par logique floue ». Presses polytechnique et universitaires Romandes. Lausanne 1994.

[**14] YAN, JUN, RYAN :** « using fuzzy logic : towards intelligent systems » Prentice Hall 1994.

[15] B.ADDAD : « Application d'une Commande Floue à Apprentissage Optimisée par des Algorithmes Génétiques au Pilotage Latéral et Longitudinal Automatique d'un Avion », PFE ENP 2007.

[16] J.FAUCHER : « Les plans d'expériences pour le réglage de commandes à base de logique floue », Thèse Doctorat INPT Toulouse. Septembre 2006.

[17] PEDRYCZ.W: « fuzzy control and fuzzy systems » Edition Research studies press 1992.

[18] HANSELMAN, DUAN, BENJAMINE : «Matlab Tools for control system analysis and design » Edition Prentice Hall 1995.

[19] David M. ALLARD: « A Multi-Objective Genetic Algorithm to Solve Single machine scheduling problems using a fuzzy fitness function ». Thèse Master de science. College of Engineering and technology of Ohio University. Juin 2007

[20] B.AMGHAR, A.HEDID: « Commande par Fuzzy Sliding Mode d'un variateur de vitesse hydrodynamique » PFE ENP 2007.

[21] FREDERIC SUR: «Présentation de la logique floue ». Mémoire de 1ère année ENS Cachan 1998.

[22] M. Passino, Stephen Yurkovich: «Fuzzy Control». Edition Addison Wesley Longman, Inc 1998.

[23] Timothy J. Ross: «Fuzzy logic with engineering applications». Edition John Wiley and Sons (Chichester) 2004.

[24] Malik LOUDINI: «contribution à la modélisation et à la commande intelligente d'un bras de rebot manipulateur flexible». Thèse doctorat d'état. ENP. Octobre 2007.

[25] A. COLEY: «An introduction to genetic algorithms for scientists and engineers ».Edition World Scientific Publishing 1999.

[26] R. Ghanea-Hercock: «Applied evolutionary algorithms in Java».Edition Springer©-London Berlin 2003.

[27] R. Ghanea-Hercock: «Soft computing: integrating evolutionary, neural, and fuzzy systems».Edition Springer©-London Berlin 2003.

[28] Hojjet Adli, Kamal.C. Sarma: «Cost Optimization of structures: Fuzzy logic, genetic algorithms, and parallel computing».Edition John Wiley & Sons Ltd 2006.

[29] M.Griebel,M.A. Schweitzer : «Meshfree Methods for Partial Differential Equations IV».Edition Springer-Verlag Berlin Heidelberg 2008.

[30] Randall J. LeVeque: «Finite Difference Methods for Ordinary and Partial Differential Equations».Edition Society for Industrial and Applied Mathematics 2007.

[31] R. Rajesh and M.R. Kaimal: «GAVLC: GA with Variable Length Chromosome for the Simultaneous Design and Stability Analysis of T-S Fuzzy Controllers».2008 IEEE International Conference on Fuzzy Systems (FUZZ 2008).

[32] M. Mucientes, D.L. Moreno, A. Bugarı´n, S. Barro: «Design of a fuzzy controller in mobile robotics using genetic algorithms». ScienceDirect, Applied Soft Computing 7 (2007) 540–546.

[33] Patrick Siarry: «Mathématique pour l'optimisation difficile». Eyrolles, Paris 2003.

[34] Pascal Rebreyend: « Algorithme génétique hybrides en optimisation combinatoire ». Thèse de Doctorat ENS Lyon 1999.

[35] Ahmed Maidi, Moussa Diaf, Jean-Pierre Corriou: «Optimal linear PI fuzzy controller design of a heat exchanger». ScienceDirect, Chemical Engineering and Processing, 47 (2008) 938–945.

[36] Maurice CLERC: «L'Optimisation par essaims particulaires ».Hermes Science 2005.

[37] Ali Allahverdi, Fawaz S. Al-Anzi: «A PSO and a Tabu search heuristics for the assembly scheduling problem of the two-stage distributed database application».Publication ScienceDirect© 2006.

[38] Maurice CLERC, Patrick SIARRY: « Une nouvelle métaheuristique pour l'optimisation difficile: la méthode des essaims particulaires ». Article de recherche, France Telecom R&D ET l'université de PARIS 12 VAL-DE-MARNE.

[39] Y.COOREN: «Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique ». Thèse Doctorat PARIS 12 VAL-DE-MARNE, 27 Novembre 2008.

[40] Felix T.S.CHAN, Manoj Kumar TIWARI: «Swarm Intelligence Focus on Ant and Particle Swarm Optimization».I-Tech Education and Publishing. Décembre 2007.

[41] Christian BLUM, Daniel MERKLE: «Swarm Intelligence introduction and applications». Springer©-Verlag Berlin Heidelberg 2008.

[42] Nadia NEDJAH, Luiza de Macedo MOURELLE: «Swarm Intelligence systems». Springer©-Verlag Berlin Heidelberg 2006. (Netherlands).

[43] James Kennedy, Russell C. Eberhart: «Swarm Intelligence». Edition Academic Press 2001.

[44] P.D Christofides: «Non linear and Robust Control of PDE Systems: Methods and Application to Transport-Reaction Processes». Edition Birkhäuser, Boston, 2001.

[45] P.D Christofides: «Control of complex process systems». International Journal of Robust and Nonlinear Control, 14(2): 87-88, 2004.

[46] Pierre Yves GLORENNEC : «Algorithmes d'apprentissage pour systèmes d'inférence floue». HERMES science publication – Paris 1999.

[47] Irène Charon : « Méthodes d'optimisation combinatoire », Edition Masson, Paris.

Résumé :

Le travail présenté dans ce mémoire porte sur l'automatisation de la synthèse de lois de commande floues en appliquant des techniques d'optimisation. Les techniques auxquelles on a fait appel sont les algorithmes génétiques et l'optimisation par essaims particulaires. Cette stratégie de synthèse et de commande a été appliquée sur la colonne d'absorption du CO_2, qui est un système complexe, à paramètres répartis, fortement non linéaire. L'évaluation des performances des contrôleurs flous optimaux obtenus, s'est faite en effectuant plusieurs tests et simulations en boucle fermée.

Mots clés : Commande floue, Optimisation, Algorithmes génétiques, optimisation par essaims particulaires (PSO), colonne d'absorption, système à paramètres répartis.

Abstract:

The work presented in this paper focuses on the automation of a fuzzy law control using two optimization techniques: genetic algorithms and particle swarm optimization. This strategy of synthesis and control was applied on a CO_2 absorption column which is a complex, highly nonlinear distributed parameters system. The evaluation of the performances of the optimal controllers was done with performing several closed-loop tests and simulations.

Keywords: Fuzzy control, Optimization, Genetic Algorithms, particle swarm optimization, absorption column, distributed parameters system.

ملخص :

العمل المعروض في هذه المذكرة يتمحور حول تحسين آلية نظام التحكم بالمنطق الغامض باستخدام طريقتين اثنين هما الخوارزميات الوراثية و البحث بطريقة الأسراب الجزيئية، و كتطبيق عملي اخترنا أسطوانة الامتصاص الذي يعتبر نظام معقد غير خطي ذو وسائط موزعة. إذن أنظمة تحكم غامضة تم تحصيلها وفق هذه الطريقة كذا تطبيقها للتحكم في هذه الأسطوانة. تقييم نجاعة هذا العمل كان بإحداث عدة اختبارات و تمثيلات في الحلقة المغلقة.

كلمات مفتاحية : التحكم بالمنطق الغامض، البحث عن الحل الأمثل، الخوارزميات الوراثية، البحث بطريقة الأسراب الجزيئية (PSO)، أسطوانة الامتصاص، نموذج ذات وسائط موزعة.