



FACULTE DES SCIENCES

U.F.R Sciences & Techniques : S.T.M.I.A

Ecole Doctorale : Informatique-Automatique-Electrotechnique-Electronique-Mathématique

Département de Formation Doctorale : Electrotechnique-Electronique

Thèse

Présentée pour l'obtention du titre de

Docteur de l'Université Henri Poincaré, Nancy-I

en Génie Electrique

par **Lotfi BAGHLI**



Contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques



Soutenue publiquement le 14 Janvier 1999 devant la commission d'examen :

Membres du Jury :

| | | |
|---------------|---------------|---|
| Président : | H. BUYSE | Professeur, LEI, Louvain-la-Neuve, Belgique |
| Rapporteurs : | J. FAUCHER | Professeur, LEEI, INPT - ENSEEIHT, Toulouse |
| | J. P. HAUTIER | Professeur, L2EP, ENSAM, Lille |
| Examineurs : | A. REZZOUG | Professeur, IUFM de Lorraine, UHP, Nancy |
| | H. RAZIK | Maître de Conférences, IUFM de Lorraine, UHP, Nancy |

| | |
|--|------------|
| Introduction générale | 5 |
| Chapitre I : Logique floue, réseaux de neurones et algorithmes génétiques | 11 |
| 1. Introduction | 13 |
| 2. Logique floue | 13 |
| 2.1. Principe et définitions | 13 |
| 2.2. Opérateurs et normes | 15 |
| 2.3. Inférence | 16 |
| 2.4. Structure d'un régulateur flou | 20 |
| 3. Réseaux de neurones | 29 |
| 3.1. Principe et définitions | 29 |
| 3.2. Perceptrons multicouches | 30 |
| 3.3. Réseaux de neurones à fonction de base radiale (RBF) | 35 |
| 3.4. Applications | 37 |
| 4. Algorithmes génétiques | 37 |
| 4.1. Principe et définitions | 37 |
| 4.2. Applications | 38 |
| 5. Conclusion | 40 |
| Chapitre II : Application à la commande de la machine asynchrone | 41 |
| 1. Introduction | 43 |
| 2. Identification des paramètres de la machine asynchrone | 43 |
| 2.1. Essai 1 : démarrage | 43 |
| 2.2. Essai 2 : échelon | 51 |
| 2.3. Discussion | 55 |
| 2.4. Conclusion | 57 |
| 3. Contrôle vectoriel classique | 58 |
| 3.1. Introduction | 58 |
| 3.2. Modèle de la machine asynchrone | 58 |
| 3.3. Méthodes de commande vectorielle des moteurs asynchrones | 61 |
| 4. Régulation, méthodes classiques | 65 |
| 4.1. Introduction | 65 |
| 4.2. Découplage | 65 |
| 4.3. Régulation des courants | 67 |
| 4.4. Régulation de la vitesse | 71 |
| 4.5. Conclusion | 79 |
| 5. Régulation par logique floue | 79 |
| 5.1. Introduction | 79 |
| 5.2. Régulateur flou à trois ensembles | 79 |
| 5.3. Régulateur flou à cinq ensembles | 87 |
| 5.4. Stabilité des systèmes intégrant un régulateur flou | 89 |
| 5.5. Conclusion | 90 |
| 6. Régulation par réseau de neurones | 95 |
| 6.1. Introduction | 95 |
| 6.2. Réseau de neurones du type perceptron | 95 |
| 6.3. Régulateur neuro-flou (à cinq ensembles flous) | 98 |
| 6.4. Réseau de neurones à fonction de base radiale (RBF) | 101 |
| 6.5. Conclusion | 105 |
| 7. Conclusion | 105 |

| | |
|---|------------|
| Chapitre III : Influence des changements de paramètres de la machine | 107 |
| 1. Introduction | 109 |
| 2. Paramètres mécaniques | 109 |
| 2.1. Moment d'inertie (J) | 109 |
| 2.2. Coefficient de frottement sec (a_{30}) | 113 |
| 3. Paramètres électriques | 115 |
| 3.1. Résistance statorique (R_s) | 115 |
| 3.2. Coefficient de dispersion (σ) | 117 |
| 3.3. Constante de temps statorique (τ_s) | 119 |
| 3.4. Constante de temps rotorique (τ_r) | 120 |
| 4. Conclusion | 121 |
| Chapitre IV : Machine avec défauts au rotor | 123 |
| 1. Introduction | 125 |
| 2. Modèle à mailles | 125 |
| 2.1. Calcul des inductances | 125 |
| 2.2. Mise en équation | 127 |
| 3. Simulation de rupture de barres rotoriques | 132 |
| 3.1. Analyse par FFT du courant statorique en régime quasi stationnaire | 135 |
| 3.2. Conclusion | 137 |
| 4. Commande vectorielle de la machine présentant des barres défectueuses | 139 |
| 4.1. Introduction | 139 |
| 4.2. Mise en œuvre | 139 |
| 4.3. Comparaison des régulateurs | 141 |
| 4.4. Conclusion | 143 |
| Chapitre V : Commande sans capteur mécanique | 145 |
| 1. Introduction | 147 |
| 2. Les méthodes de commande sans capteur mécanique | 147 |
| 2.1. Méthodes à base d'estimateur | 147 |
| 2.2. Méthodes à base d'observateur | 148 |
| 2.3. Commande directe du couple (DTC) | 150 |
| 2.4. Système adaptatif utilisant un modèle de référence (MRAS) | 153 |
| 2.5. Filtre de Kalman | 155 |
| 2.6. Autres méthodes | 158 |
| 3. Méthode proposée | 159 |
| 3.1. Commande vectorielle sans capteur avec régulation de vitesse | 167 |
| 4. Techniques "intelligentes" de commande sans capteur | 170 |
| 5. Conclusion | 172 |
| Conclusion générale | 173 |
| Annexes | 179 |
| 1. Annexe 1 : Paramètres des machines étudiées | 181 |
| 1.1. Machine 1 | 181 |
| 1.2. Machine 2 | 181 |
| 2. Annexe 2 : Présentation du logiciel MASVECT | 182 |
| 3. Annexe 3 : Description du dispositif expérimental | 185 |

Table des matières

Nomenclature _____ **189**

Glossaire _____ **193**

Bibliographie _____ **197**

Introduction générale

La machine asynchrone, de par sa construction, est la machine la plus robuste et la moins chère du marché. Les progrès réalisés en commande et les avancées technologiques considérables, tant dans le domaine de l'électronique de puissance que dans celui de la micro-électronique, ont rendu possible l'implantation de commandes performantes de cette machine faisant d'elle un concurrent redoutable dans les secteurs de la vitesse variable et du contrôle rapide du couple.

Cependant, de nombreux problèmes demeurent. L'influence des variations des paramètres de la machine, le comportement en fonctionnement dégradé, la présence d'un capteur mécanique sont autant de difficultés qui ont aiguisé la curiosité des chercheurs dans les laboratoires. En témoignage, le nombre sans cesse grandissant des publications qui traitent le sujet.

Bien que déjà présents dans d'autres domaines, la logique floue, les réseaux de neurones et les algorithmes génétiques constituaient, au début de notre travail de thèse, une nouveauté dans le domaine de l'électrotechnique.

Nous avons voulu savoir quel pourrait être l'apport de ces méthodes appliquées à l'identification et la commande de la machine asynchrone.

Des questions se posent alors naturellement : ces méthodes qui ne relèvent, en tout cas pas à première vue, d'une logique cartésienne classique, peuvent-elles conduire à de meilleurs résultats ? Sinon, que présentent-elles comme avantages et inconvénients par rapport aux techniques conventionnelles ?

Ce qui nous a le plus motivé dans ce travail, c'est qu'il comporte de nombreux volets et touche à plusieurs disciplines en même temps. Il comporte par-dessus tout une partie expérimentale forte en enseignements et qui nous permet, non seulement de voir concrètement l'aboutissement et la finalité de l'étude, mais aussi de faire ressortir les problèmes cruciaux de mise en œuvre.

Il est évidemment impossible de cerner toutes les possibilités et les combinaisons où peuvent intervenir ces techniques. Nous examinerons plus précisément la régulation ainsi que son optimisation.

Il nous apparaît nécessaire de commencer par présenter ces méthodes puis d'explicitier les manières les plus simples de les mettre en œuvre.

Tout au long de ce travail, nous avons gardé comme objectif l'implantation et l'expérimentation des méthodes développées. En effet, combien de procédés donnent de très bons résultats en simulation et ne fonctionnent pas du tout dès qu'il s'agit de les utiliser concrètement. Nous essayerons aussi d'éviter les procédés trop complexes pour une implantation raisonnable ou qui ne permettent pas de tirer des conclusions quant aux performances obtenues.

L'aspect expérimental revêt donc une très grande importance. Il permet de valider les méthodes établies par un travail théorique et de simulation. Il met également en évidence certaines lacunes dues à une mauvaise modélisation ou identification. Dans de nombreux cas, les résultats sont assez satisfaisants puisque la régulation pallie ces erreurs, mais il arrive que la méthode, une fois implantée, ne fonctionne pas du tout. L'expérimentation en laboratoire constitue donc un garde-fou avant l'étape d'industrialisation.

Les différents travaux concernant le sujet font l'objet de cinq chapitres qui constituent ce mémoire.

Le chapitre I présente les différentes approches utilisées dans cette étude. Ces techniques étant relativement nouvelles dans la communauté électrotechnique, il est important de bien préciser les termes employés et de développer suffisamment les méthodes utilisées.

Nous commençons par définir et expliquer la terminologie utilisée en logique floue, la théorie des ensembles flous et ainsi que le mode de raisonnement propre aux variables floues. Nous développons une méthode de synthèse d'un régulateur flou et abordons les étapes nécessaires à la réalisation de l'inférence floue.

Nous présentons par la suite les deux structures de réseaux de neurones auxquelles nous nous sommes intéressés dans ce travail. Les méthodes et propriétés de l'apprentissage des réseaux de neurones sont également abordées.

Enfin, une méthode d'optimisation par algorithme génétique est présentée. Les notions, directement inspirées de la théorie de l'évolution, qui interviennent dans ces algorithmes sont explicitées.

Dans ce chapitre, l'accent est mis aussi bien sur la conception des méthodes que sur leur utilisation. Ceci afin de se familiariser avec une manière de raisonner différente par rapport à l'approche à laquelle on est habitué dans la résolution de certains problèmes.

L'application de ces méthodes fait l'objet du chapitre II. En premier lieu, nous présentons l'identification de la machine asynchrone par algorithmes génétiques. Les méthodes basées sur l'essai de démarrage de la machine et sur des échelons de tension continue sont étudiées. Une analyse du comportement de l'algorithme ainsi que de la précision des méthodes est menée.

Par la suite, la commande vectorielle de la machine asynchrone ainsi que les modèles utilisés pour la simulation de la machine et pour le contrôle sont présentés. Une comparaison des différentes méthodes sera abordée de manière plus détaillée dans le chapitre V.

Le fonctionnement et le réglage des différents contrôleurs du schéma classique sont explicités. A chaque fois, les résultats expérimentaux et de simulation sont comparés et commentés.

On s'intéresse alors au remplacement du régulateur classique de vitesse, au sein de la commande vectorielle, par un régulateur flou puis par un régulateur neuronal. Les régulateurs flous à trois puis à cinq ensembles flous par variable sont comparés par rapport aux régulateurs classiques du type Proportionnel-Intégral.

Les régulateurs neuronaux de type perceptron sont ensuite utilisés afin de dupliquer la surface de commande des régulateurs flous. La procédure d'apprentissage, propre à cette structure, ainsi que son utilisation sont discutées.

Dans un premier temps, afin d'éviter cette procédure d'apprentissage qui constitue une étape lourde en terme de temps de calcul, des régulateurs neuronaux à fonction de base radiale (RBF) sont mis en œuvre. Une amélioration de leurs performances, par optimisation de leurs caractéristiques à l'aide d'algorithmes génétiques, est effectuée et abouti au régulateur RBF retenu pour la suite de l'étude.

Les résultats obtenus suite à des essais lorsque des échelons de vitesse sont appliqués à vide et en charge ont permis de tirer des conclusions constructives.

Dans le chapitre III, nous présentons une étude, par simulation, sur l'influence des changements de paramètres de la machine sur la réponse du système. Les différents régulateurs classiques, flous et neuronaux précédemment retenus sont utilisés et les résultats obtenus sont comparés. Les essais portent aussi bien sur des échelons de vitesse que sur des échelons de couple résistant. Là encore, le comportement des régulateurs est analysé suivant

leurs surfaces caractéristiques et une confrontation des résultats obtenus permet de compléter les conclusions du chapitre II sur les performances des régulateurs.

Dans le chapitre IV, nous abordons le problème de la machine asynchrone présentant des défaillances de structure. Nous nous intéressons, plus spécialement, à la rupture de barres rotoriques et à son impact sur le fonctionnement de la machine aussi bien en régime non commandé qu'en contrôle vectoriel pour une régulation de vitesse. Nous verrons comment les régulateurs envisagés perçoivent ces défauts en régime transitoire et en régime quasi permanent.

Le chapitre V présente le problème de la commande vectorielle de la machine asynchrone sans capteur mécanique. Un état de l'art est dressé et les différentes méthodes sont comparées et analysées. La solution que nous recherchons va dans le sens de la simplicité d'implantation et de la robustesse vis à vis des variations de paramètres. Une estimation du flux statorique par intégration de la f.e.m. est alors proposée. L'accent est mis sur l'influence des décalages des signaux de mesures, aussi minimes soient ils, sur le fonctionnement du dispositif. La mise au point d'une méthode de compensation d'offset en ligne, rend possible l'utilisation de ce procédé. Des essais expérimentaux montrent l'efficacité d'une telle commande de couple.

Nous terminons par une conclusion sur l'ensemble de cette étude et nous proposons des perspectives de travail.

Les annexes contiennent les paramètres des machines étudiées, une présentation du logiciel que nous avons développé au cours de cette étude et du dispositif expérimental élaboré. Des problèmes d'ordre pratique sont également abordés dans cette partie.

Chapitre I

*Logique floue, réseaux de neurones
et algorithmes génétiques*

1. Introduction

On pourrait dire que la logique floue, les réseaux de neurones et les algorithmes génétiques constituent des approches qui, tout compte fait, ne sont pas nouvelles. Leur développement se fait à travers les méthodes par lesquelles l'homme essaye de copier la nature et de reproduire des modes de raisonnement et de comportement qui lui sont propres. Bien que ces approches paraissent "naturelles", et si elles se sont imposées dans des domaines allant du traitement de l'image à la gestion financière, elles commencent à peine à être utilisées dans les domaines de l'électrotechnique et de l'industrie afin de résoudre les problèmes d'identification, de régulation de processus, d'optimisation, de classification, de détection de défauts ou de prise de décision.

Considérant que la machine asynchrone à cage et le convertisseur statique associé posent des problèmes difficiles à étudier pour sa commande, nous nous proposons d'analyser ce que les méthodes décrites peuvent apporter comme solution à cette commande. Il est évidemment impossible de cerner toutes les possibilités et les combinaisons où elles peuvent intervenir dans un tel processus. Nous examinerons plus précisément la régulation ainsi que son optimisation.

Il nous apparaît nécessaire de commencer par présenter ces méthodes puis d'explicitier les manières les plus simples de les mettre en œuvre. Plutôt que développer des méthodes trop générales, nous cadrerons notre travail de façon à répondre à l'objectif fixé ici, la commande vectorielle de la machine asynchrone, et d'analyser les avantages et les inconvénients liés à ces méthodes.

2. Logique floue

2.1. Principe et définitions

La logique floue repose sur la théorie des ensembles flous développée par Zadeh [ZAD 65]. A coté d'un formalisme mathématique fort développé, nous préférons aborder la présentation de manière intuitive.

Les notions de température moyenne ou de courant faible sont relativement difficiles à spécifier de manière précise. On peut fixer des seuils et considérer que l'on attribue tel ou tel qualificatif en fonction de la valeur de la variable par rapport à ces seuils. Ceci ne peut exprimer qu'un avis très tranché du qualificatif "température moyenne" par exemple. L'aspect "vague" de ce qualificatif n'est pas représenté (figure 1.1).

On peut définir le degré d'appartenance de la variable température à l'ensemble "faible" comme le "degré de vérité" de la proposition "la température est faible".

En logique booléenne, le degré d'appartenance (μ) ne peut prendre que deux valeurs (0 ou 1). La température peut être :

- faible : $\mu_{faible}(T) = 1, \mu_{moyenne}(T) = 0, \mu_{élevée}(T) = 0$
- moyenne : $\mu_{faible}(T) = 0, \mu_{moyenne}(T) = 1, \mu_{élevée}(T) = 0$
- élevée : $\mu_{faible}(T) = 0, \mu_{moyenne}(T) = 0, \mu_{élevée}(T) = 1$

Elle ne peut pas prendre deux qualificatifs à la fois.

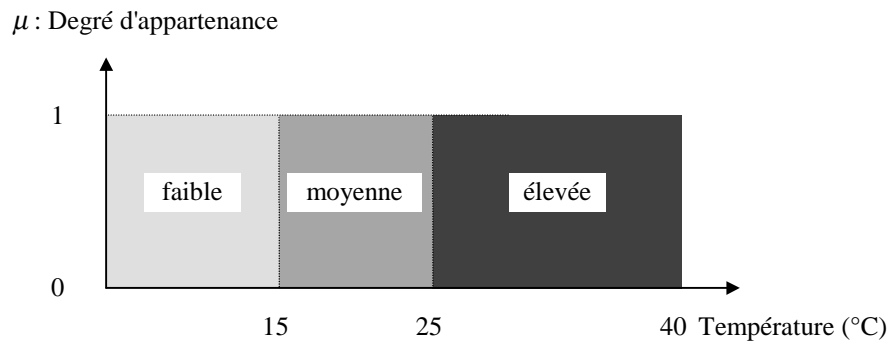


Figure 1.1 Exemple d'ensembles considérés en logique booléenne

En logique floue, le degré d'appartenance devient une fonction qui peut prendre une valeur réelle comprise entre 0 et 1 inclus.

$\mu_{moyenne}(T)$, par exemple, permet de quantifier le fait que la température puisse être considérée comme moyenne.

Dans ce cas, la température peut être considérée, à la fois, comme faible avec un degré d'appartenance de 0,2 et comme moyenne avec un degré d'appartenance de 0,8 (figure 1.2).

$$\mu_{faible}(T) = 0,2, \quad \mu_{moyenne}(T) = 0,8, \quad \mu_{élevée}(T) = 0$$

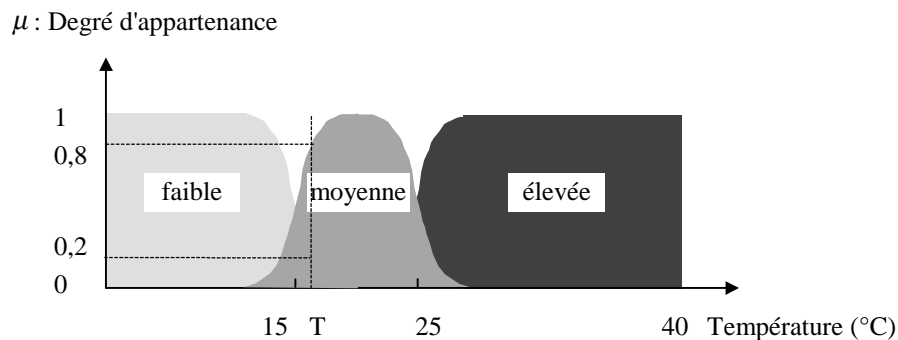


Figure 1.2 Exemple d'ensembles considérés en logique floue

Pour la variable floue x , on définit un ensemble flou A sur un univers de discours X par une fonction degré d'appartenance :

$$\begin{aligned} \mu_A : X &\rightarrow [0,1] \\ x &\mapsto \mu_A(x) \end{aligned} \tag{1.1}$$

L'univers de discours est l'ensemble des valeurs réelles que peut prendre la variable floue x et $\mu_A(x)$ est le degré d'appartenance de l'élément x à l'ensemble flou A (figure 1.3).

Plus généralement, le domaine de définition de $\mu_A(x)$ peut être réduit à un sous-ensemble de X [ZAD 65]. On peut ainsi avoir plusieurs fonctions d'appartenance, chacune caractérisant un sous-ensemble flou. C'est par l'association de tous les sous-ensembles flous de l'univers de discours, que l'on obtient l'ensemble flou de la variable floue x [MAM 75]. Par abus de langage, les sous-ensembles flous sont fort souvent confondus avec l'ensemble flou.

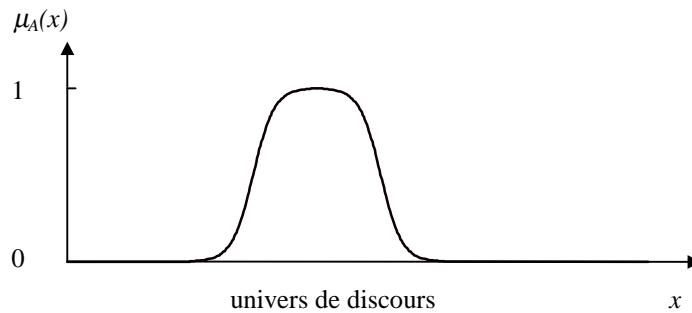


Figure 1.3 Représentation d'un ensemble flou par sa fonction d'appartenance

Dans notre exemple, la variable floue est la température, l'univers de discours est l'ensemble des réels de l'intervalle $[0, 40]$. On attribue à cette variable trois sous-ensembles flous : faible, moyenne et élevée. Chacun est caractérisé par sa fonction degré d'appartenance : $\mu_{faible}(T)$, $\mu_{moyenne}(T)$ et $\mu_{élevée}(T)$.

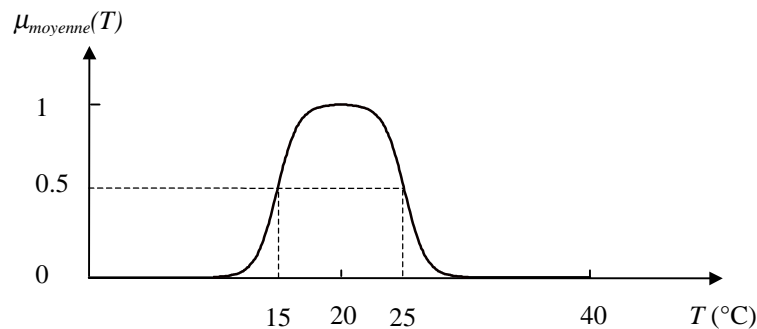


Figure 1.4 Cas de l'ensemble flou "moyenne" de la variable Température

On peut définir la fonction degré d'appartenance $\mu_{moyenne}$ sur tout l'univers de discours :

$$\mu_{moyenne}(x) = \begin{cases} \frac{1}{1 + \exp(15 - x)}; & x \in [0, 20] \\ 1 - \frac{1}{1 + \exp(25 - x)}; & x \in [20, 40] \end{cases} \quad (1.2)$$

2.2. Opérateurs et normes

Comme dans la théorie des ensembles classiques, on définit l'intersection, l'union des ensembles flous ainsi que le complémentaire d'un ensemble flou. Ces relations sont traduites par les opérateurs "et", "ou" et "non". De nouvelles fonctions d'appartenance liées à ces opérateurs sont établies :

$$x \text{ appartient à } A \text{ et } B \Leftrightarrow x \in A \cap B \Leftrightarrow \mu_{A \cap B}(x)$$

$$x \text{ appartient à } A \text{ ou } B \Leftrightarrow x \in A \cup B \Leftrightarrow \mu_{A \cup B}(x)$$

$$x \text{ appartient au complément de } A \Leftrightarrow x \in \bar{A} \Leftrightarrow \mu_{\bar{A}}(x)$$

L'opérateur "et" se définit par une norme triangulaire (t-norme) :

$$T : [0,1] \times [0,1] \rightarrow [0,1]$$

$$(x, y) \mapsto z = xTy$$

T possède les propriétés suivantes :

- commutativité : $xTy = yTx$
- associativité : $xT(yTz) = (xTy)Tz$
- monotonie : $xTz \leq yTz$ si $x \leq y$
- admet 0 comme élément absorbant et 1 comme élément neutre : $0Tx = 0$, $1Tx = x$

De même, l'opérateur "ou" se définit par une co-norme triangulaire (T^*) qu'on appelle aussi s-norme (S) :

$$S : [0,1] \times [0,1] \rightarrow [0,1]$$

$$(x, y) \mapsto z = xSy$$

S possède également les propriétés de commutativité, d'associativité et de monotonie. Elle admet 1 comme élément absorbant et 0 comme élément neutre.

A l'aide de la loi de Morgan, on peut associer à chaque t-norme, la s-norme définie par :

$$xSy = 1 - (1 - x)T(1 - y).$$

Les opérateurs les plus utilisés en logique floue sont :

- L'opérateur "et" pour la t-norme, qui correspond à l'intersection de deux ensembles A et B . Il peut être réalisé par :
- La fonction "Min" : $\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$
- La fonction arithmétique "Produit" : $\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x)$
- L'opérateur "ou" pour la s-norme, qui correspond à l'union de deux ensembles A et B . Il peut être réalisé par :
- La fonction "Max" : $\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$
- La fonction arithmétique "Somme" : $\mu_{A \cup B}(x) = \mu_A(x) + \mu_B(x)$
- L'opérateur "non" est réalisé par : $\mu_{\bar{A}}(x) = 1 - \mu_A(x)$.

2.3. Inférence

En logique classique, la règle de raisonnement du *modus ponens* permet, à partir des deux assertions,

- x est A
- et
- si x est A alors y est B ,
- de conclure que y est B .

En logique floue, la règle s'appelle *modus ponens généralisé* et permet à partir des assertions,

- x est A'

et

- si x est A alors y est B ,

de conclure que y est B' .

L'inférence est l'opération d'agrégation des règles.

Sans entrer dans les détails de formalisation mathématique, qui se basent sur les notions de sous-ensembles flous, de graphes (Γ est le graphe définissant la relation (R) de A vers B) et de projection (B' est la projection sur B de A' par le graphe Γ), il est possible de définir l'ensemble B' par :

$$\forall y \in B, \mu_{B'}(y) = \sup_{x \in A} \mu_{A' \times B \cap \Gamma}(x, y).$$

C'est à dire que le degré d'appartenance de chaque élément y de B à l'ensemble flou B' est égal au plus grand degré d'appartenance des couples (x, y) à l'intersection de l'ensemble A' avec le graphe Γ de la relation R .

Ce dernier est calculé en utilisant la fonction "Min" pour l'opérateur "et" de l'intersection :

$$\mu_{A' \times B \cap \Gamma}(x, y) = \min(\mu_{A'}(x), \mu_R(x, y)).$$

En ce qui nous concerne, nous allons nous intéresser aux inférences avec plusieurs règles. En effet, dans le cas de la commande et de la régulation, les variables floues ont plusieurs ensembles d'appartenance. Ainsi plusieurs règles peuvent être activées en même temps.

Les règles d'inférences peuvent être décrites de plusieurs façons,

a) Linguistiquement :

On écrit les règles de façon explicite comme dans l'exemple suivant,

SI (la température est élevée **ET** la vitesse est faible) **ALORS** la tension est grande positive

OU

SI (la température est moyenne **ET** la vitesse est faible) **ALORS** la tension est positive

b) Symboliquement :

Il s'agit en fait d'une description linguistique où l'on remplace la désignation des ensembles flous par des abréviations.

c) Par matrice d'inférence :

Elle rassemble toutes les règles d'inférences sous forme de tableau. Dans le cas d'un tableau à deux dimensions, les entrées du tableau représentent les ensembles flous des variables d'entrées (température : T et vitesse : V). L'intersection d'une colonne et d'une ligne donne l'ensemble flou de la variable de sortie définie par la règle. Il y a autant de cases que de règles.

Exemple :

| U | | T | | |
|-----|---|-----|---|----|
| | | F | M | E |
| V | F | Z | P | GP |
| | E | Z | Z | P |

Tableau 1.1

Les règles que décrit ce tableau sont (sous forme symbolique) :

SI (T est F **ET** V est F) **ALORS** $U=Z$

OU

SI (T est M **ET** V est F) **ALORS** $U=P$

OU

SI (T est E **ET** V est F) **ALORS** $U=GP$

OU

SI (T est F **ET** V est E) **ALORS** $U=Z$

OU

SI (T est M **ET** V est E) **ALORS** $U=Z$

OU

SI (T est E **ET** V est E) **ALORS** $U=P$

Dans l'exemple ci-dessus, on a représenté les règles qui sont activées à un instant donné par des cases sombres :

SI (T est M **ET** V est F) **ALORS** $U=P$

OU

SI (T est E **ET** V est F) **ALORS** $U=GP$

Il arrive que toutes les cases du tableau ne soient pas remplies, on parle alors de règles d'inférences incomplètes. Cela ne signifie pas que la sortie n'existe pas, mais plutôt que le degré d'appartenance est nul pour la règle en question.

Il s'agit maintenant de définir les degrés d'appartenance de la variable de sortie à ses sous-ensembles flous. Nous allons présenter les méthodes d'inférence qui permettent d'y arriver.

Ces méthodes se différencient essentiellement par la manière dont vont être réalisés les opérateurs (ici "ET" et "OU") utilisés dans les règles d'inférence.

Nous présentons les trois méthodes d'inférence les plus usuelles ; Max-Min, Max-Produit et Somme-Produit :

2.3.1. Méthode d'inférence Max-Min

Cette méthode réalise l'opérateur "ET" par la fonction "Min", la conclusion "ALORS" de chaque règle par la fonction "Min" et la liaison entre toutes les règles (opérateur "OU") par la fonction Max.

La dénomination de cette méthode, dite Max-Min ou "implication de Mamdani", est due à la façon de réaliser les opérateurs ALORS et OU de l'inférence.

Reprenons l'exemple précédent où seulement deux règles sont activées :

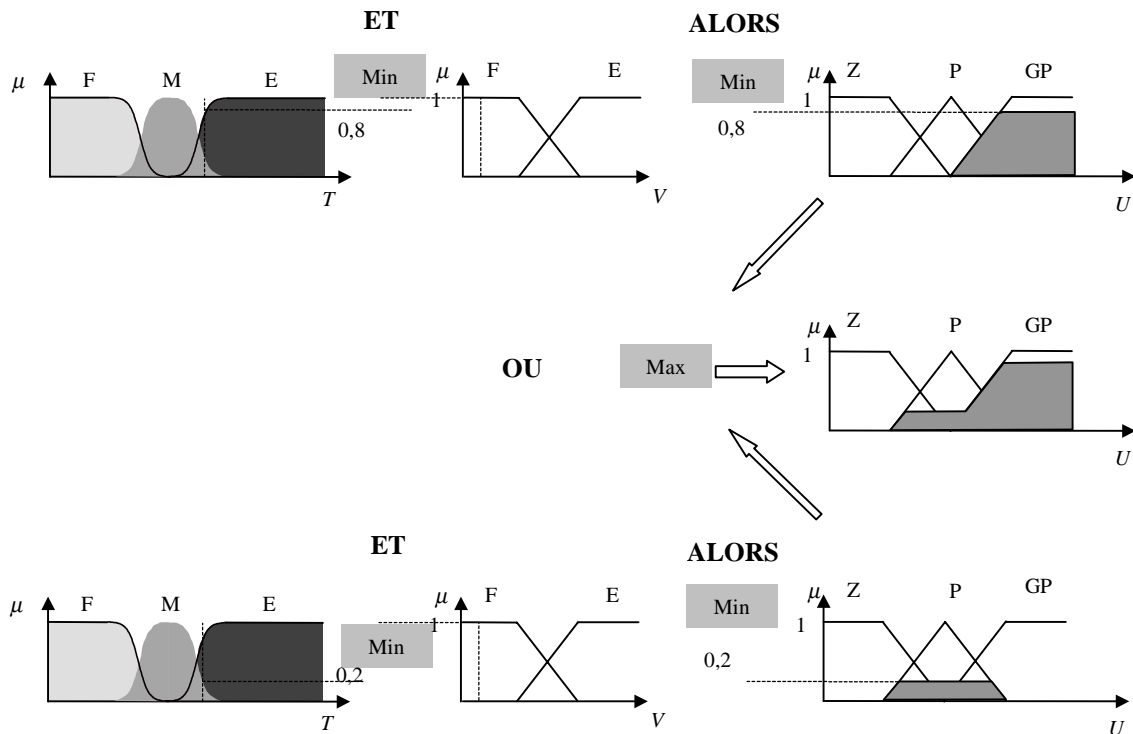


Figure 1.5 Exemple d'inférence Max-Min

La variable T est Elevée avec un degré d'appartenance de 0,8 et Moyenne avec un degré d'appartenance de 0,2. La vitesse V est faible avec un degré d'appartenance de 1. L'application de la première règle d'inférence donne un degré d'appartenance à la condition de 0,8 (minimum dû à l'opérateur ET entre les deux degrés d'appartenance). On obtient ainsi une "fonction d'appartenance partielle" dessinée en gris qui est écrêtée à 0,8. De manière similaire, la seconde règle donne lieu à une fonction d'appartenance écrêtée à 0,2.

La fonction d'appartenance résultante correspond au maximum des deux fonctions d'appartenance partielles puisque les règles sont liées par l'opérateur OU.

2.3.2. Méthode d'inférence Max-Produit

La différence par rapport à la méthode précédente réside dans la manière de réaliser la conclusion "ALORS". Dans ce cas, on utilise le produit comme illustré par la figure 1.6.

On remarque que les fonctions d'appartenances partielles ici ont la même forme que la fonction d'appartenance dont elles sont issues multipliées par un facteur d'échelle vertical qui correspond au degré d'appartenance obtenu à travers l'opérateur "ET".

On l'appelle également "implication de Larsen".

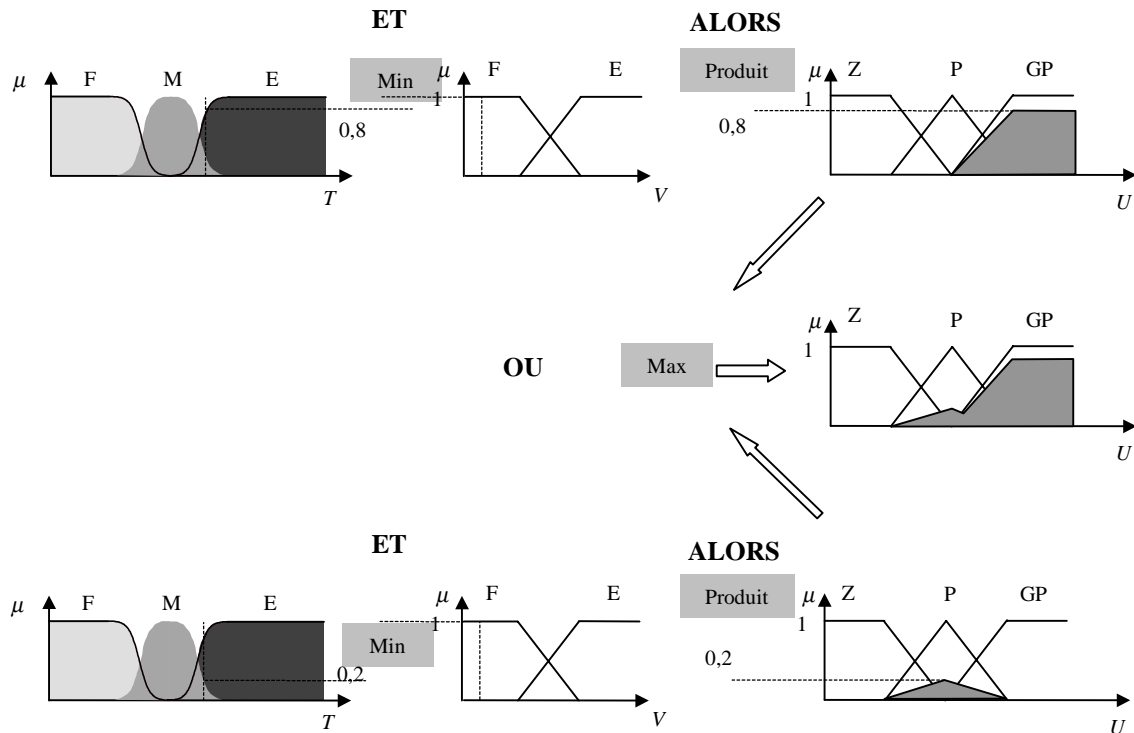


Figure 1.6 Exemple d'inférence Max-Produit

2.3.3. Méthode d'inférence Somme-Produit

Dans ce cas, l'opérateur "ET" est réalisé par le produit, de même que la conclusion "ALORS". Cependant, l'opérateur "OU" est réalisé par la valeur moyenne des degrés d'appartenance intervenant dans l'inférence.

D'autres méthodes ont été élaborées, ayant chacune une variante spécifique. Néanmoins, la méthode Max-Min est de loin la plus utilisée à cause de sa simplicité.

2.4. Structure d'un régulateur flou

2.4.1. Introduction

Après avoir énoncé les concepts de base et les termes linguistiques utilisés en logique floue, nous présentons la structure d'un contrôleur flou.

La réalisation d'un régulateur flou pose un problème lié aux nombreuses manières de réaliser les opérateurs flous et l'implication. Bien que la liste des méthodes présentées ne soit pas exhaustive, on se retrouve déjà avec un grand nombre de variantes. Nous allons présenter les principales différences entre ces dernières et justifier, sinon expliquer, nos choix. L'aboutissement à une implantation expérimentale constitue le but à atteindre après les étapes de modélisation et de simulation.

Dans ce qui suit, nous nous intéressons principalement au régulateur de vitesse au sein d'une commande vectorielle de la machine asynchrone [BAG 96b].

La vitesse de référence peut être pilotée par un opérateur externe. La grandeur de sortie de ce régulateur de vitesse est l'image du couple électromagnétique de référence que l'ensemble commande-convertisseur-machine doit générer. A flux constant, ce couple est proportionnel au courant I_{qs}^* (courant I_{qs} de référence) imposé en entrée à la boucle de régulation de courant.

Le schéma de base du régulateur repose sur la structure d'un régulateur classique à la différence que l'on va retenir la forme incrémentale. Cette dernière donne en sortie, non pas le couple ou le courant à appliquer mais plutôt l'incrément de cette grandeur.

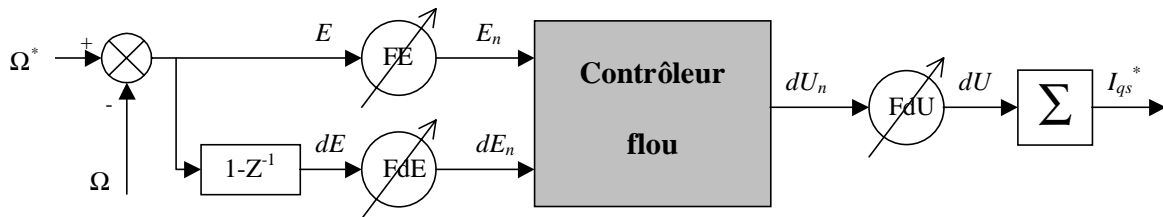


Figure 1.7 Schéma synoptique d'un régulateur flou de vitesse

Dans le schéma ci-dessus comme dans ce qui suit, nous notons :

E : l'erreur, elle est définie par :

$$E(k) = \Omega^*(k) - \Omega(k) \quad (1.3)$$

dE : la dérivée de l'erreur, elle est approchée par :

$$dE(k) = \frac{E(k) - E(k-1)}{T_e}, \quad T_e \text{ étant la période d'échantillonnage.} \quad (1.4)$$

La sortie du régulateur est donnée par :

$$I_{qs}^*(k) = I_{qs}^*(k-1) + dU(k) \quad (1.5)$$

On retrouve en entrée et en sortie du contrôleur flou des gains dits "facteurs d'échelle" qui permettent de changer la sensibilité du régulateur flou sans en changer la structure. Les grandeurs indicées "n" sont donc les grandeurs normalisées à l'entrée et à la sortie du contrôleur flou.

Les règles d'inférences permettent de déterminer le comportement du contrôleur flou. Il doit donc inclure des étapes intermédiaires qui lui permettent de passer des grandeurs réelles vers les grandeurs floues et vice versa ; ce sont les étapes de fuzzification et de defuzzification (figure 1.8).

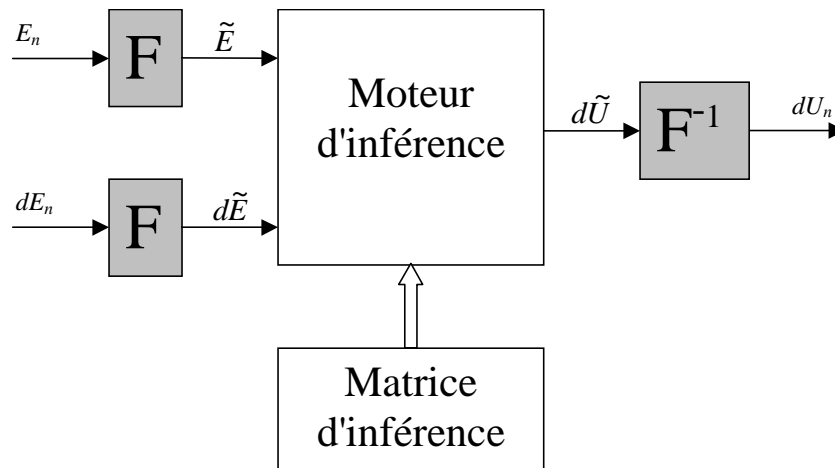


Figure 1.8 Structure du régulateur flou

2.4.2. Fuzzification

Les ensembles flous des variables d'entrée et leurs fonctions d'appartenance sont à définir en premier lieu.

L'étape de fuzzification permet de fournir les degrés d'appartenance de la variable floue à ses ensembles flous en fonction de la valeur réelle de la variable d'entrée.

Le choix du nombre des ensembles flous, de la forme des fonctions d'appartenance, du recouvrement des ces fonctions et de leur répartition sur l'univers de discours n'est pas évident. Il y a cependant des facteurs qui sont plus important que d'autres (cf. § 2.4.5).

Une subdivision très fine de l'univers de discours sur plus de sept ensembles flous n'apporte en général aucune amélioration du comportement dynamique du système à réguler [BUH 94]. Par contre, on peut obtenir des comportements non linéaires assez différents en fonction de la manière dont les fonctions d'appartenance des ensembles flous sont disposées sur l'univers de discours.

Nous avons opté pour des fonctions triangulaires et trapézoïdales pour les variables d'entrées (figure 1.9). Elles permettent une implantation facile et l'étape de fuzzification ne requiert alors que peu de temps de calcul lors de son évaluation en temps réel.

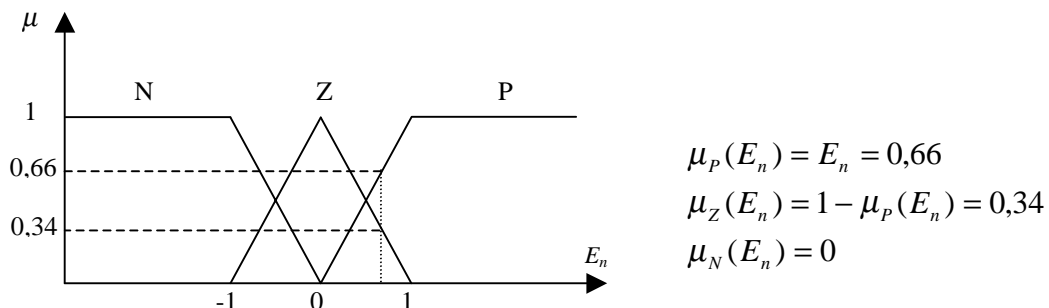


Figure 1.9 Fuzzification de l'erreur

Le recouvrement de deux fonctions d'appartenance voisines est de 1 ; c'est à dire que leur croisement s'effectue à $\mu=0,5$. Un recouvrement insuffisant voir inexistant conduit à une zone où aucune règle d'inférence n'est sollicitée. De même, un recouvrement trop important, surtout avec un degré d'appartenance près de l'unité, conduit à un aplatissement de la caractéristique du régulateur [BUH 94]. Le passage d'une fonction d'appartenance à sa voisine doit s'effectuer en douceur de manière à ce qu'il y ait au moins deux règles d'inférences qui soient sollicitées en même temps.

Il est également indispensable de pouvoir fuzzifier la variable de sortie. En effet, lors de l'inférence et de la defuzzification, on a besoin de connaître les ensembles flous de cette variable ainsi que leurs fonctions d'appartenance.

Du point de vue implantation, [MAM 75] utilise une table de fuzzification donnant les degrés d'appartenance aux ensembles flous pour des valeurs discrètes que prend la variable sur l'univers de discours.

2.4.3. Inférence

Comme nous l'avons précédemment évoqué, nous allons nous baser sur une matrice ou table d'inférence pour cette étape.

La construction d'une telle table d'inférence repose sur une analyse qualitative du processus. Dans notre cas c'est une analyse dans le plan de phase de la trajectoire que l'on souhaite donner au système.

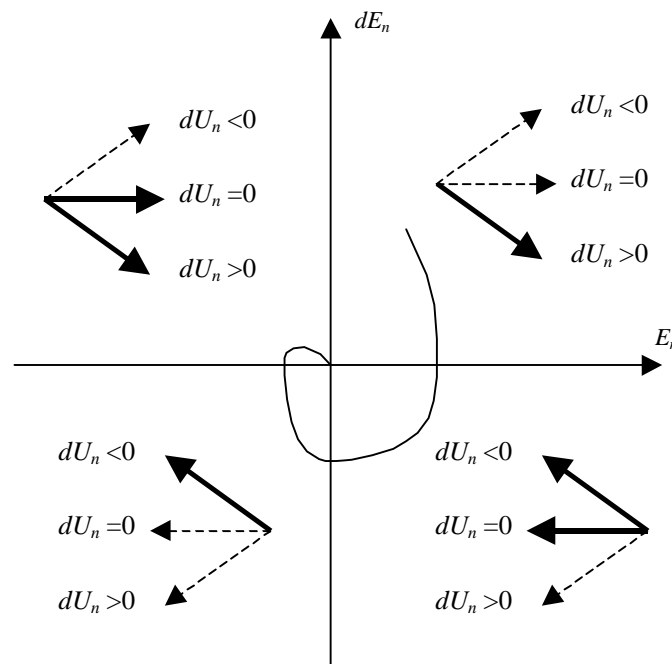


Figure 1.10 Trajectoire dans le plan de phase

Une action dans un sens ou dans l'autre de la commande provoque le déplacement dans une direction indiquée par les flèches (figure 1.10). En pointillés sont indiquées les directions que l'on ne souhaite pas donner au système car il serait alors divergent.

Si l'on attribue trois ensembles flous, Négatif, Zéro et Positif, à chacune des variables floues, on peut écrire pour chaque quadrant le comportement à adopter.

Par exemple :

Quadrant 1 :

SI E_n est P **ET** dE_n est P **ALORS** dU_n est P

Quadrant 2 :

SI E_n est N **ET** dE_n est P **ALORS** dU_n est Z

On remarque cependant que dans le cas de ce quadrant, on pourrait souhaiter donner un comportement différent en sollicitant la commande de manière à freiner l'annulation de l'erreur ; on pourrait tout aussi bien choisir comme règle :

SI E_n est N **ET** dE_n est P **ALORS** dU_n est N

Si l'on ne retient que trois ensembles flous et deux entrées, la matrice d'inférence est la suivante :

| | | dE_n | | |
|-------|---|--------|---|---|
| | | N | Z | P |
| E_n | N | N | N | Z |
| | Z | N | Z | P |
| | P | Z | P | P |

Tableau 1.2

Si par contre, on subdivise l'univers de discours avec plus d'ensembles flous et/ou qu'on prenne en compte la dérivée seconde (sdE_n), alors le choix des règles devient plus vaste et son optimisation dépend du système à réguler.

Dans le cas où l'on attribue cinq ensembles flous (GN, N, Z, P et GP)¹ aux variables, un choix possible est le suivant :

| | | dE_n | | | | |
|-------|----|--------|----|---|----|----|
| | | GN | N | Z | P | GP |
| E_n | GN | GN | GN | N | N | Z |
| | N | GN | N | N | Z | P |
| | Z | N | N | Z | P | P |
| | P | N | Z | P | P | GP |
| | GP | Z | P | P | GP | GP |

Tableau 1.3

¹ G signifie "Grand".

Ou

| dU_n | | dE_n | | | | |
|--------|----|--------|----|----|----|----|
| | | GN | N | Z | P | GP |
| E_n | GN | GN | GN | GN | GN | Z |
| | N | GN | N | N | Z | P |
| | Z | N | N | Z | P | P |
| | P | N | Z | P | P | GP |
| | GP | Z | GP | GP | GP | GP |

Tableau 1.4

Celle qui suit ne possède que huit règles qui donnent un incrément de commande non nul (différent de l'ensemble flou Z). Elle convient à la régulation de processus du type 1^{er} ordre [RAM 93].

| dU_n | | dE_n | | | | |
|--------|----|--------|---|----|---|----|
| | | GN | N | Z | P | GP |
| E_n | GN | Z | Z | GN | Z | Z |
| | N | Z | Z | N | Z | Z |
| | Z | N | N | Z | P | P |
| | P | Z | Z | P | Z | Z |
| | GP | Z | Z | GP | Z | Z |

Tableau 1.5

Si de plus, on ne retient que les règles donnant un ensemble flou différent de Z, le processus d'inférence se retrouve sensiblement allégé. C'est un exemple de table d'inférence incomplète.

L'inférence se fait donc sur la base des matrices que l'on vient de décrire. On commence par utiliser un opérateur t-norme pour définir la description symbolique associée à la prémisse de la règle ; C'est à dire réaliser le "ET". On passe ensuite à l'inférence proprement dite qui consiste à caractériser la variable floue de sortie pour chaque règle. C'est l'étape de la conclusion "ALORS".

Enfin, la dernière étape de l'inférence, appelée agrégation des règles, permet de synthétiser ces résultats intermédiaires. On utilise une s-norme.

Comme nous l'avons vu, la manière de réaliser les opérateurs va donner lieu à des contrôleurs flous différents. Les régulateurs les plus courants sont ceux de :

- Mamdani :
Ces contrôleurs sont dits symboliques car la prémisse et la conclusion sont symboliques [MAM 75] [MAM 76]. Après l'inférence, il faut passer par une étape de "defuzzification" afin d'obtenir la valeur réelle de la commande à appliquer.
- Sugeno :
Ils sont dits de type procédural [TAK 83]. En effet, seule la prémisse est symbolique. La conclusion, qui correspond à la commande, est directement une constante réelle ou une expression polynomiale.

L'établissement des règles d'inférence est généralement basé sur un des points suivants [TAK 83] :

- L'expérience de l'opérateur et/ou du savoir-faire de l'ingénieur en régulation et contrôle.
- Un modèle flou du processus pour lequel on souhaite synthétiser le régulateur.
- Les actions de l'opérateur ; s'il n'arrive pas à exprimer linguistiquement les règles qu'il utilise implicitement.
- L'apprentissage ; c'est dire que la synthèse de règle se fait par un procédé automatique également appelé superviseur. Souvent, des réseaux neuronaux y sont associés.

L'évaluation des règles d'inférence étant une opération déterministe, il est tout à fait envisageable de mettre sous forme de tableau ce contrôleur.

Il reste, toutefois, intéressant dans certains systèmes complexes, de garder l'approche linguistique plutôt que d'avoir à faire à un nombre trop important de valeurs précises [KIN 77].

De plus, un algorithme linguistique peut être examiné et discuté directement par quelqu'un qui n'est pas l'opérateur mais qui possède de l'expérience sur le comportement du système.

La formulation linguistique de la sortie permet également d'utiliser le régulateur flou en boucle ouverte donnant ainsi à l'opérateur les consignes à adopter.

Si, après inférence, on se retrouve avec un ensemble flou de sortie caractérisé par l'apparition de plus d'un maximum, cela révèle l'existence d'au moins deux règles contradictoires (figure 1.11). Une grande zone plate (figure 1.12), moins grave de conséquence, indiquerait que les règles, dans leur ensemble, sont faibles et mal formulées.

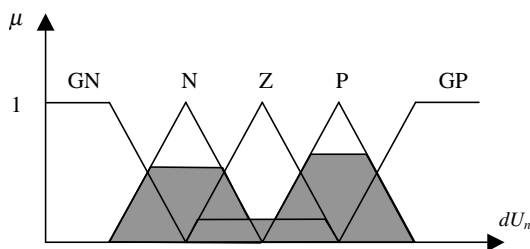


Figure 1.11

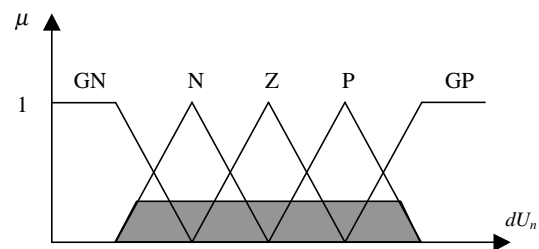


Figure 1.12

2.4.4. Defuzzification

Par cette étape se fait le retour aux grandeurs de sortie réelles. Il s'agit de calculer, à partir des degrés d'appartenance à tous les ensembles flous de la variable de sortie, l'abscisse qui correspond à la valeur de cette sortie. Différentes méthodes sont utilisées :

- Méthode du centre de gravité :
C'est la méthode de defuzzification la plus courante. L'abscisse du centre de gravité de la fonction d'appartenance résultant de l'inférence correspond à la valeur de sortie du régulateur.

$$dU_n = \frac{\int x\mu_R(x)dx}{\int \mu_R(x)dx} \quad (1.6)$$

Il apparaît que plus la fonction d'appartenance résultante est compliquée, plus le processus de defuzzification devient long et coûteux en temps de calcul.

- Méthode par valeur maximum :
Cette méthode est beaucoup plus simple. La valeur de sortie est choisie comme l'abscisse de la valeur maximale de la fonction d'appartenance.

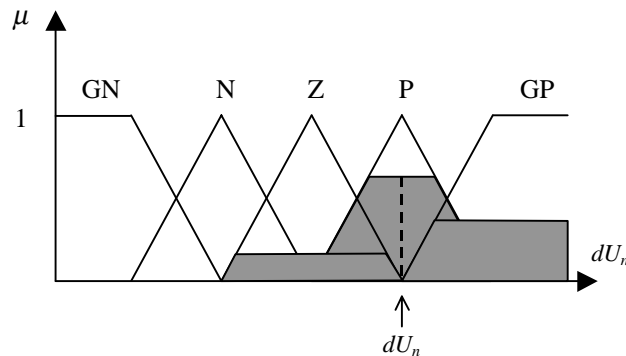


Figure 1.13 Defuzzification par valeur maximum

- Méthode des hauteurs pondérées :
Elle correspond à la méthode de centre de gravité quand les fonctions d'appartenance ne se recouvrent pas.

$$dU_n = \frac{\sum x\mu_{R_i}(x)}{\sum \mu_{R_i}(x)} \quad (1.7)$$

Cette méthode est surtout utilisée quand les fonctions d'appartenance de la variable de sortie sont des singletons.

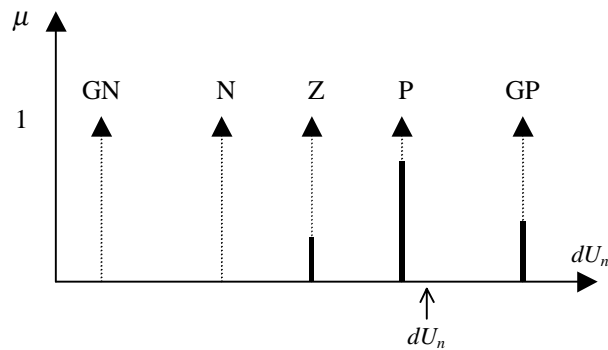


Figure 1.14 Defuzzification par la méthode des hauteurs pondérées

Dans ce cas, le calcul du centre de gravité se trouve grandement simplifié. Le régulateur n'est plus de type Mamdani mais de type Sugeno de part la façon dont la sortie est calculée.

Le régulateur flou à deux entrées est représenté par sa surface caractéristique (figure 1.15). Cette dernière exprime les variations de la valeur réelle de la sortie du régulateur en fonction des entrées quand ces dernières parcourent l'univers de discours.

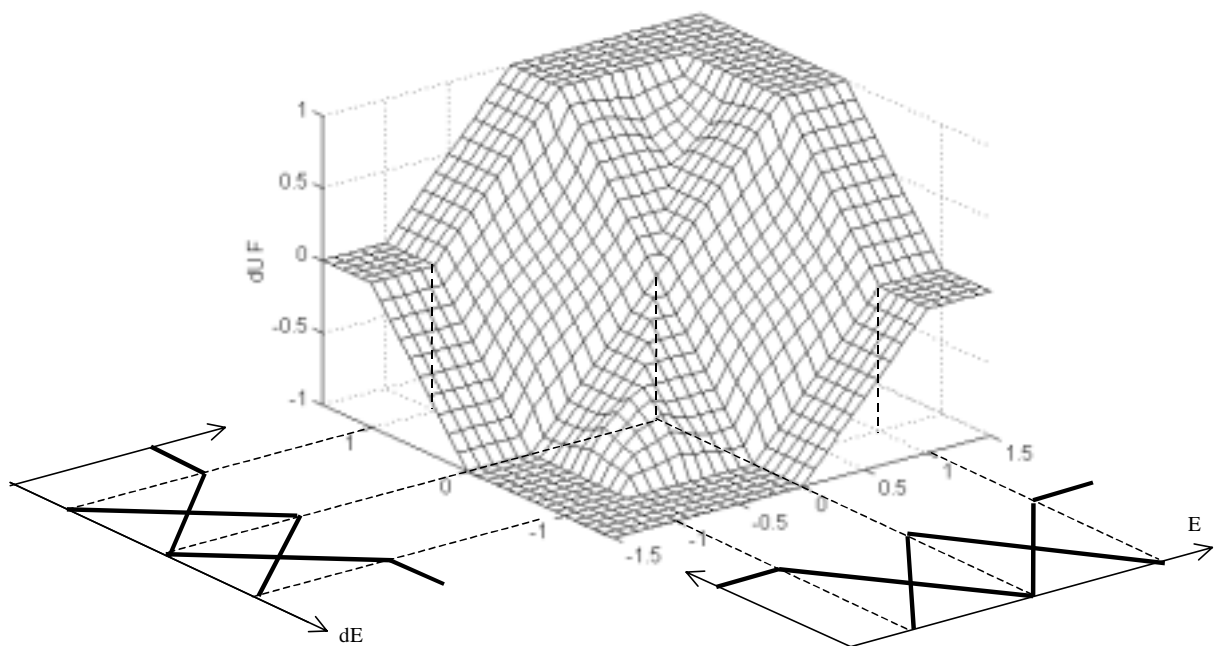


Figure 1.15 Surface caractéristique d'un régulateur flou

2.4.5. Conclusion

L'obtention d'un contrôle performant requiert une bonne formulation des règles. Dans notre cas, l'analyse dans le plan de phase permet de dégager rapidement un tableau de règles correct. Ceci n'est pas le cas pour des systèmes non-linéaires complexes où le modèle est très compliqué sinon inexistant. Il convient également d'accorder une grande importance au processus présentant des temps morts. Un exemple d'association d'un estimateur flou à un

régulateur PI flou est donné dans [AOK 90]. Il a permis de prendre en compte le retard caractéristique d'un four destiné à la fonte du verre.

D'un point de vue pratique, on peut résumer dans les points suivants les éléments qui ont peu d'importance sur le comportement global du régulateur flou [LUT 96] :

- La forme des fonctions d'appartenance, d'où le choix de formes triangulaires à cause de la simplicité de mise en œuvre.
- Le choix des fonctions pour réaliser les opérateurs et le mécanisme d'inférence (agrégation des règles : ALORS)
- Le choix de la méthode de defuzzification (hauteurs, C.G.)

Par contre, il faut faire prêter une attention particulière, lors de la synthèse du régulateur flou :

- Au nombre et surtout à la répartition des fonctions d'appartenance sur l'univers de discours.
- A la table des règles.

Ces considérations pratiques ont guidé le choix développé au chapitre II.

3. Réseaux de neurones

3.1. Principe et définitions

L'origine des réseaux de neurones vient de l'essai de modélisation du neurone biologique par Warren McCulloch et Walter Pitts [JOD 94]. Ils supposent que l'impulsion nerveuse est le résultat d'un calcul simple effectué par chaque neurone et que la pensée née grâce à l'effet collectif d'un réseau de neurones interconnectés.

Le schéma suivant présente un neurone formel :

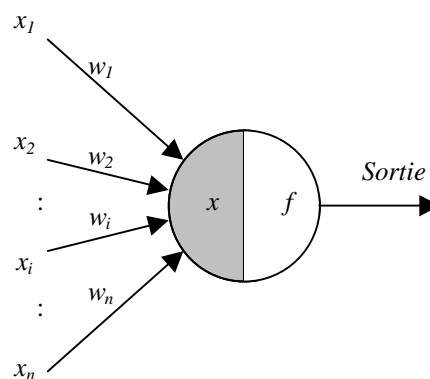


Figure 1.16 Représentation d'un neurone

Les entrées du neurone sont désignées par x_i ($i=1..n$). Les paramètres w_i reliant les entrées aux neurones sont appelés poids synaptiques ou tout simplement poids. La sortie du neurone est donnée par :

$$\text{Sortie} = f(x) \text{ avec } x = \sum_{i=1}^n w_i x_i \quad (1.8)$$

x est l'état d'activation du neurone (également appelé état ou activité).

f est la fonction d'activation du neurone. Conformément au modèle biologique, les fonctions d'activation sont généralement croissantes et bornées. Les fonctions les plus connues sont la fonction signe, la fonction linéaire saturée et la fonction sigmoïde.

Leur choix revêt une importance capitale comme nous le verrons par la suite.

Les réseaux de neurones sont constitués des neurones élémentaires connectés entre eux par l'intermédiaire des poids qui jouent le rôle des synapses. L'information est portée par la valeur de ces poids tandis que la structure du réseau de neurones ne sert qu'à traiter cette information et à l'acheminer vers la sortie.

Le réseau de neurones fait partie des Réseaux Adaptatifs Non-linéaires, cela signifie que ses agents (neurones) s'organisent et modifient leurs liens mutuels lors d'une procédure fondamentale qu'est l'apprentissage. Pour une tâche précise, l'apprentissage du réseau de neurones consiste donc à adapter les différents poids w_i .

3.2. Perceptrons multicouches

Ce sont les réseaux de neurones les plus connus. Un perceptron est un réseau de neurones artificiel du type *feedforward*, c'est à dire à propagation directe. Sur l'exemple suivant nous présentons un perceptron à trois couches. La première est celle des entrées (elle n'est cependant pas considérée comme couche neuronale par certains auteurs car elle est linéaire et ne fait que distribuer les variables d'entrées). La deuxième est dite couche cachée (ou couche intermédiaire) et constitue le cœur du réseau de neurones. Ses fonctions d'activation sont du type sigmoïde. La troisième, constituée ici par un seul neurone est la couche de sortie. Sa fonction d'activation est du type linéaire bornée.

Nous pouvons remarquer sur la figure 1.17, des termes x_0^m en entrée des neurones ². En fait, sur chaque neurone, en plus de ses entrées qui les lient avec les neurones précédents, on ajoute une entrée particulière que l'on appelle polarisation du neurone. Elle correspond à un biais qui joue un rôle de translation du domaine d'activité du neurone. Sa valeur est donc liée à la fonction d'activation puisqu'elle permet le déplacement de cette fonction.

Afin de garder une notation généralisée, nous représentons ces biais comme le produit d'une entrée x_0^m par les poids w_{0j}^m . Nous fixons l'entrée x_0^m à l'unité, le poids porte alors l'information sur la polarisation du neurone.

Le perceptron multicouche est très utilisé en identification et en contrôle. Avec une couche cachée, il constitue un approximateur universel. De récentes recherches montrent qu'il peut être entraîné de manière à approximer n'importe quelle fonction entrées-sorties sous réserve de mettre suffisamment de neurones dans la couche cachée et d'utiliser des sigmoïdes pour les fonctions d'activation [PIC 94]. Bien entendu, les théorèmes mathématiques ne démontrent pas qu'un réseau à une seule couche cachée est optimal [THI 97].

² Les termes en exposant représentent, non pas la fonction puissance, mais plutôt l'indice (m) de la couche du réseau de neurones.

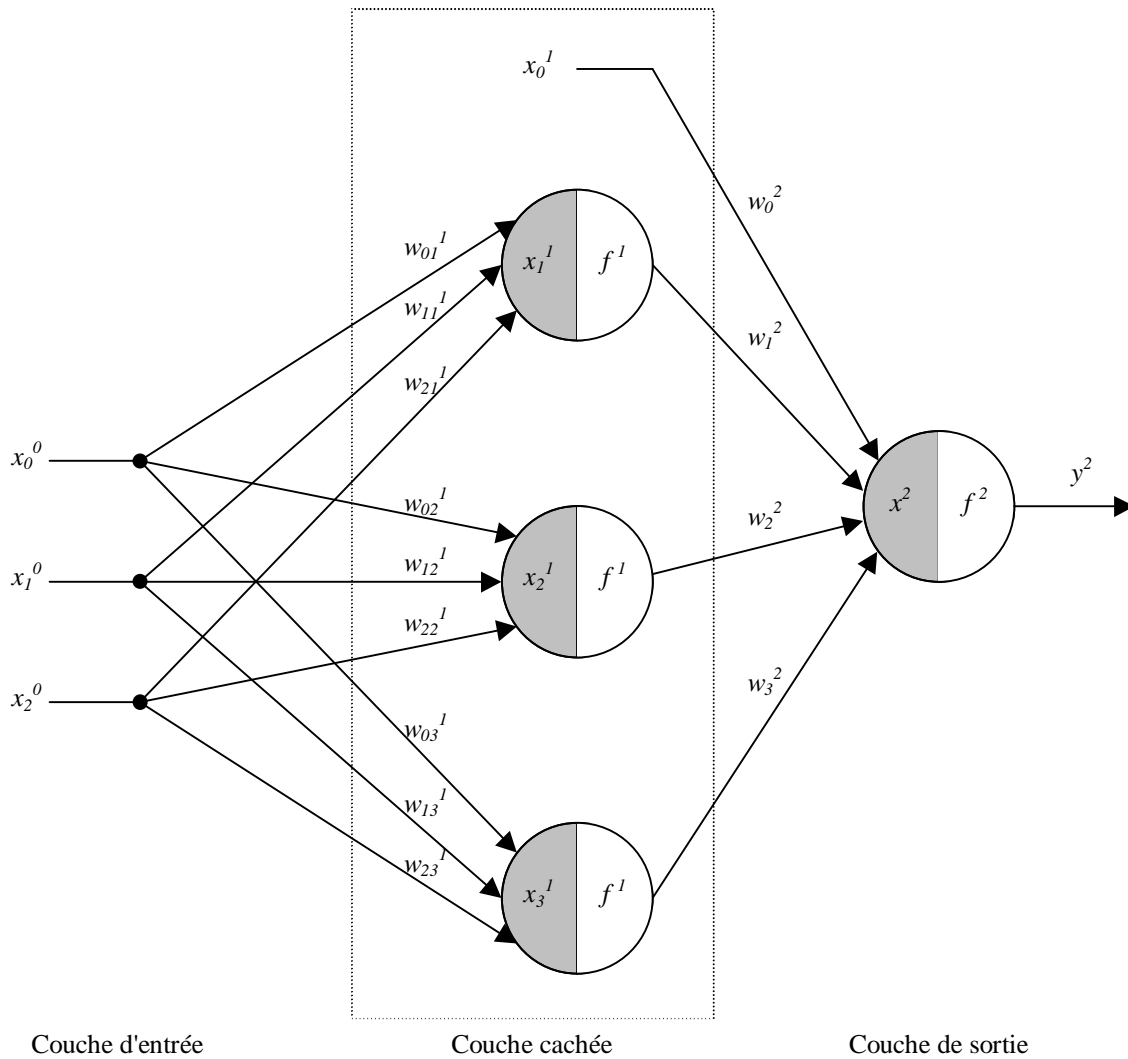


Figure 1.17 Réseau de neurones de type perceptron à une couche cachée

L'un des problèmes de l'utilisation des réseaux de neurones consiste dans le choix de sa topologie. Par exemple, il n'existe pas de règle générale qui donne le nombre de neurones à retenir pour la couche intermédiaire. Ce choix est spécifique à chaque application et, à ce jour, ce ne sont que des choix arbitraires dont on vérifie par la suite la validité.

3.2.1. Apprentissage

Une fois la structure fixée, il faut passer par le processus d'apprentissage, par lequel les poids vont être ajustés de manière à satisfaire un critère d'optimisation.

Prenons le cas de l'identification d'un processus qui comporte deux entrées et une sortie. L'apprentissage va se faire sur un ensemble de triplet (x_1^0, x_2^0, y^0) .

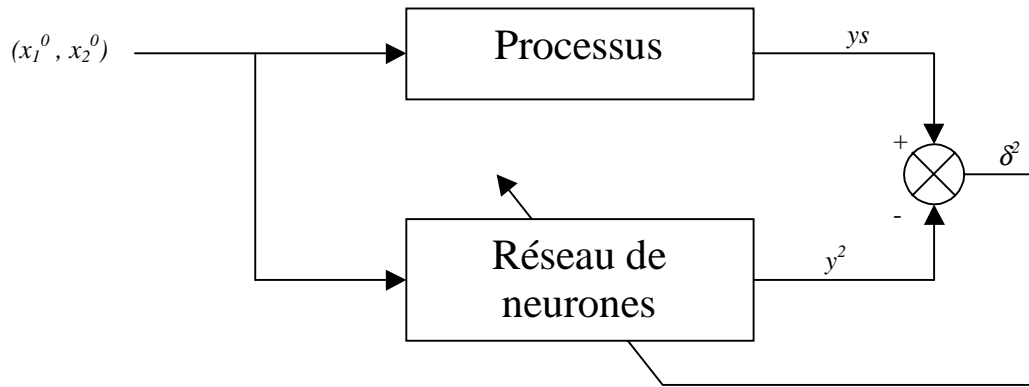


Figure 1.18 Schéma synoptique du procédé d'apprentissage du réseau de neurones

Pour chaque triplet, l'erreur entre les deux sorties est calculée. Elle est utilisée pour corriger les poids de la couche de sortie puis par *rétropropagation de l'erreur* (*error backpropagation*), des erreurs intermédiaires, correspondant à la couche cachée sont ainsi calculées et permettent l'ajustement des poids w_{ij}^1 de la couche cachée.

Nous présentons, dans ce qui suit, un exemple de cet algorithme. Il est basé sur la méthode du gradient. La notation adoptée peut être résumée comme suit :

Les entrées x_i^0 sont notées y_i^0 pour permettre de généraliser l'algorithme.

La fonction d'activation pour la sortie est $f(x)=x$.

$$x_j^m = \sum_i w_{ij}^m y_i^{m-1} \text{ représente l'activité du neurone } j \text{ de la couche } m. \quad (1.9)$$

$$y_j^m = f(x_j^m) \text{ est la sortie du neurone } j \text{ de la couche } m. \quad (1.10)$$

w_{ij}^m est le poids reliant la sortie du neurone (i) de la couche précédente ($m-1$) au neurone (j) de la couche considérée (m).

i, j, i', j', k et l sont des indices muets.

L'erreur globale sur l'ensemble d'apprentissage (indicé (k)) est : ³

$$E = \frac{1}{2} \sum_k (y^{S(k)} - y^{2(k)})(y^{S(k)} - y^{2(k)}) \quad (1.11)$$

Les poids vont être ajustés par une variation de Δw_i^2 et Δw_{ij}^1 . La direction optimale est donnée par l'opposé du gradient de l'erreur par rapport aux poids :

$$\begin{aligned} \Delta w_i^2 &= -\eta \frac{\partial E}{\partial w_i^2} = -\frac{1}{2} \eta \sum_k \frac{\partial}{\partial w_i^2} \{ (y^{S(k)} - y^{2(k)})(y^{S(k)} - y^{2(k)}) \} \\ &= \eta \sum_k (y^{S(k)} - y^{2(k)}) \frac{\partial y^{2(k)}}{\partial w_i^2} \end{aligned}$$

³ Afin d'éviter toute confusion entre les indices hauts et les exposants, nous écrivons l'erreur quadratique sous sa forme explicite (Erreur * Erreur).

$$\begin{aligned}\frac{\partial y^{2(k)}}{\partial w_{i'}^2} &= \frac{\partial}{\partial w_{i'}^2} \left\{ f^2 \left(\sum_{i=0}^3 w_i^2 y_i^1(k) \right) \right\} = f^{2'}(x^{2(k)}) \frac{\partial}{\partial w_{i'}^2} \left\{ \sum_{i=0}^3 w_i^2 y_i^1(k) \right\} \\ &= f^{2'}(x^{2(k)}) y_{i'}^1(k)\end{aligned}$$

$$\boxed{\Delta w_{i'}^2 = \eta \sum_k (y_{S(k)} - y^{2(k)}) f^{2'}(x^{2(k)}) y_{i'}^1(k)} \quad (1.12)$$

De même pour la couche cachée :

$$\begin{aligned}\Delta w_{i'j'}^1 &= -\eta \frac{\partial E}{\partial w_{i'j'}^1} = -\frac{1}{2} \eta \sum_k \frac{\partial}{\partial w_{i'j'}^1} \left\{ (y_{S(k)} - y^{2(k)}) (y_{S(k)} - y^{2(k)}) \right\} \\ &= \eta \sum_k (y_{S(k)} - y^{2(k)}) \frac{\partial y^{2(k)}}{\partial w_{i'j'}^1} = \eta \sum_k (y_{S(k)} - y^{2(k)}) \left\{ \sum_j \frac{\partial y^{2(k)}}{\partial x_j^1(k)} \frac{\partial x_j^1(k)}{\partial w_{i'j'}^1} \right\} \\ &= \eta \sum_k (y_{S(k)} - y^{2(k)}) \left\{ f^{2'}(x^{2(k)}) \frac{\partial}{\partial x_{j'}^1} \left\{ \sum_{i=0}^3 w_i^2 f^1(x_i^1(k)) \right\} y_{i'}^0(k) \right\}\end{aligned}$$

$$\boxed{\Delta w_{i'j'}^1 = \eta \sum_k (y_{S(k)} - y^{2(k)}) f^{2'}(x^{2(k)}) w_{j'}^2 f^1(x_{j'}^1(k)) y_{i'}^0(k)} \quad (1.13)$$

On peut introduire l'erreur élémentaire à la couche m par $\delta_j^m(k)$, on a alors :

$$\boxed{\delta^{2(k)} = (y_{S(k)} - y^{2(k)}) f^{2'}(x^{2(k)})} \quad (1.14)$$

$$\boxed{\Delta w_i^2 = \eta \sum_k \delta^{2(k)} y_i^1(k)} \quad (1.15)$$

De même

$$\boxed{\begin{aligned}\delta_j^{1(k)} &= (y_{S(k)} - y^{2(k)}) f^{2'}(x^{2(k)}) w_{j'}^2 f^1(x_{j'}^1(k)) \\ &= \delta^{2(k)} w_{j'}^2 f^1(x_{j'}^1(k))\end{aligned}} \quad (1.16)$$

$$\boxed{\Delta w_{ij}^1 = \eta \sum_k \delta_j^{1(k)} y_i^0(k)} \quad (1.17)$$

On voit bien que pour le calcul de $\delta_j^{1(k)}$, on utilise $\delta^{2(k)}$. En fait, de proche en proche, par rétropropagation, on calcule une erreur correspondant à chaque neurone pour une couche donnée. On peut remarquer cela plus explicitement sur un cas plus général où l'on a plusieurs sorties dans la dernière couche. L'erreur est alors donnée par :

$$E = \frac{1}{2} \sum_k \sum_j (y_{S_j(k)} - y_j^2(k)) (y_{S_j(k)} - y_j^2(k)) \quad (1.18)$$

Dans la dernière couche :

$$\Delta w_{i'j'}^2 = -\eta \frac{\partial E}{\partial w_{i'j'}^2} = \eta \sum_k \sum_j (y_{sj(k)} - y_j^2(k)) \frac{\partial y_j^2(k)}{\partial w_{i'j'}^2}$$

$$\frac{\partial y_j^2(k)}{\partial w_{i'j'}^2} = \frac{\partial}{\partial w_{i'j'}^2} \left\{ f^2 \left(\sum_i w_{ij}^2 y_i^1(k) \right) \right\} = \begin{cases} f^{2'}(x_j^2(k)) y_{i'}^1(k) & \text{si } j = j' \\ 0 & \text{sinon} \end{cases}$$

$$\Delta w_{i'j'}^2 = \eta \sum_k (y_{sj(k)} - y_j^2(k)) f^{2'}(x_j^2(k)) y_{i'}^1(k)$$

(1.19)

Dans l'avant-dernière couche :

$$\Delta w_{i'j'}^1 = -\eta \frac{\partial E}{\partial w_{i'j'}^1} = \eta \sum_k \sum_j (y_{sj(k)} - y_j^2(k)) \frac{\partial y_j^2(k)}{\partial w_{i'j'}^1}$$

$$= \eta \sum_k \sum_j (y_{sj(k)} - y_j^2(k)) \left\{ \sum_i \sum_l \frac{\partial y_j^2(k)}{\partial x_i^2(k)} \frac{\partial x_i^2(k)}{\partial x_l^1(k)} \frac{\partial x_l^1(k)}{\partial w_{i'j'}^1} \right\}$$

$$\Delta w_{i'j'}^1 = \eta \sum_k \sum_j (y_{sj(k)} - y_j^2(k)) f^{2'}(x_j^2(k)) w_{j'l}^2 f^{1'}(x_l^1(k)) y_{i'}^0(k)$$

(1.20)

On a alors :

$$\delta_j^2(k) = (y_{sj(k)} - y_j^2(k)) f^{2'}(x_j^2(k))$$

(1.21)

$$\Delta w_{ij}^2 = \eta \sum_k \delta_j^2(k) y_i^1(k)$$

(1.22)

De même

$$\delta_j^1(k) = \left\{ \sum_l (y_{sl(k)} - y_l^2(k)) f^{2'}(x_l^2(k)) w_{jl}^2 \right\} f^{1'}(x_j^1(k))$$

$$= \left\{ \sum_l \delta_l^2(k) w_{jl}^2 \right\} f^{1'}(x_j^1(k))$$

(1.23)

$$\Delta w_{ij}^1 = \eta \sum_k \delta_j^1(k) y_i^0(k)$$

(1.24)

On peut également montrer que la relation générale pour toute couche différente de la couche de sortie, on a :

$$\delta_j^{m-1}(k) = \left\{ \sum_l \delta_l^m(k) w_{jl}^m \right\} f^{m-1'}(x_j^{m-1}(k))$$

(1.25)

$$\Delta w_{ij}^{m-1} = \eta \sum_k \delta_j^{m-1}(k) y_i^{m-2}(k)$$

(1.26)

De ce qui précède découle un certain nombre de remarques :

- On peut choisir la vitesse avec laquelle se fait la mise à jour des poids lors de l'apprentissage en agissant sur η . Il représente un facteur d'accélération appelé ici facteur d'apprentissage.

- Une variante permet l'introduction d'un terme inertiel qui aide à la convergence [LUT 96],
$$w_{ij}^m(n) = w_{ij}^m(n-1) + \Delta w_{ij}^m(n) + \mu \cdot \Delta w_{ij}^m(n-1) \quad (1.27)$$

Le choix de ce facteur est cependant délicat, on peut d'ailleurs aboutir à des effets inverses ; des oscillations ou un ralentissement de la convergence. Les performances apportées par ce terme ne sont pas toujours convaincantes [REN 95].
- L'algorithme de rétropropagation introduit la dérivée première des fonctions d'activation. Comme nous le verrons dans le chapitre suivant, nous utiliserons pour notre application comme fonctions d'activation, la fonction sigmoïde pour la couche cachée et la fonction identité pour la sortie.
- Il est cependant tout à fait envisageable d'utiliser d'autres algorithmes qui ne nécessitent pas de dérivation. Par exemple [BUR 97] utilise un algorithme de changement aléatoire des poids. Bien que cet algorithme ne prenne pas la pente la plus grande, il y a toujours une bonne probabilité pour qu'un petit nombre d'essais aléatoires trouvent une direction dans la quelle les poids sont ajustés de manière à réduire l'erreur.
Dans [EL-S 94], les auteurs exposent une méthode génétique pour l'adaptation des poids. Nous verrons l'utilisation des algorithmes génétiques par la suite.
- Ces méthodes ont pour but de réduire les problèmes rencontrés lors de la convergence. En effet, l'algorithme du gradient est très sensible aux minimums locaux. Le choix d'un facteur d'apprentissage variable permet dans certains cas d'accélérer la convergence [PIC 94]. Il arrive cependant qu'on reste au-dessus du critère d'arrêt sans jamais l'atteindre. C'est souvent le signe que le mécanisme d'apprentissage est inadapté, ou que la topologie du réseau ne permet pas d'atteindre ce degré de précision. Dans ce cas, il faut augmenter le nombre de neurones de la couche cachée ou changer de structure.
- Villiers et al. [De V 92] pensent qu'il n'y a pas de raison d'utiliser deux couches cachées à la place d'une seule couche cachée pour une complexité de réseau ⁴ donnée. De plus, les réseaux de neurones à deux couches cachées sont plus sensibles au problème du minimum local durant l'apprentissage (utilisant les méthodes de rétropropagation de l'erreur et du gradient-conjugué).
- Un autre phénomène lié à l'algorithme de rétropropagation utilisé sous sa forme locale (la mise à jour des poids se fait au fur et à mesure du parcours de l'ensemble d'apprentissage) est que l'on se retrouve pour des surfaces symétriques ($y_s = \text{surface}(x_1^0, x_2^0)$) avec un décalage en sortie pour $(x_1^0, x_2^0) = (0, 0)$. Nous mettrons en évidence ce phénomène dans le chapitre II.

3.3. Réseaux de neurones à fonction de base radiale (RBF)

Les réseaux de neurones à fonction de base radiale (Radial Basis Functions) sont des réseaux de neurones à une seule couche cachée dont les fonctions d'activation sont des fonctions à base radiale, le plus souvent des gaussiennes. La fonction d'activation du neurone de la couche de sortie est l'identité. Les entrées sont directement connectées aux neurones de la couche cachée. La sortie d'un neurone de la couche cachée est donnée par :

⁴ La complexité du réseau de neurones peut être estimée par le nombre de poids utilisés dans l'architecture du réseau.

$$y_j^1 = f^1\left(\frac{\|X - C_j\|}{\sigma_j}\right) = \exp\left(-\frac{1}{2} \frac{\|X - C_j\|^2}{\sigma_j^2}\right) \quad (1.28)$$

$$y_j^1 = \exp\left(-\sum_i \frac{1}{2} \frac{(x_i^0 - c_{ij})^2}{\sigma_j^2}\right) \quad (1.29)$$

Et la sortie par :

$$y^2 = \sum_i w_i^2 y_i^1 \quad (1.30)$$

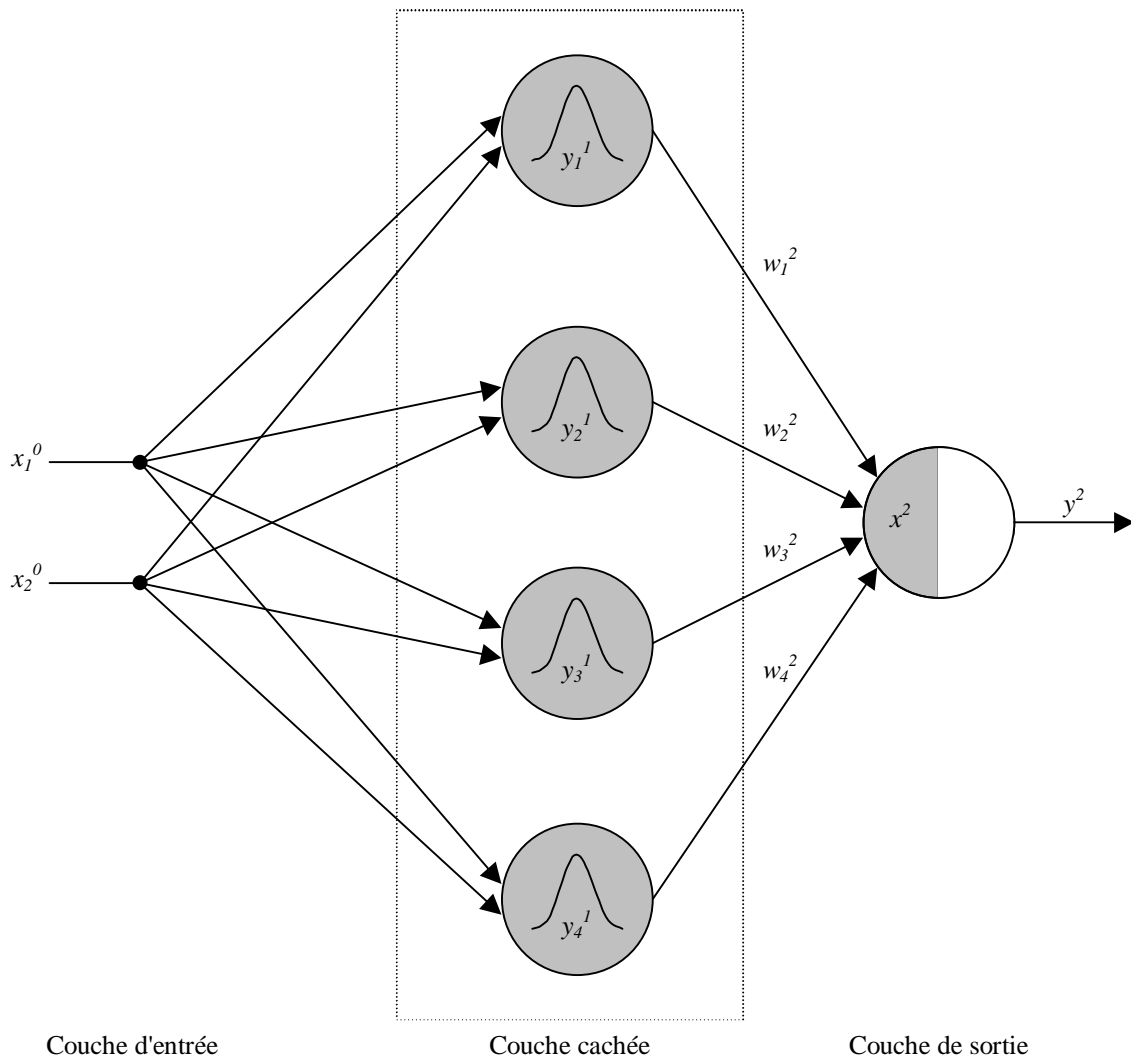


Figure 1.19 Réseau de neurones de type RBF

Les gaussiennes sont définies par leur centre c_{ij} et leur écart type σ_j . Ces derniers, avec les poids w_i^2 , sont les paramètres à optimiser en vue de l'apprentissage. Ce réseau de neurones, par rapport aux perceptrons multicouches, a comme particularité la localisation de l'excitation du neurone. En effet, un vecteur d'entrée donné ne sollicitera que les quelques neurones dont le domaine de réception (défini par les couples c_{ij}, σ_j) contient la

valeur de cette entrée. La réponse de la fonction de base radiale ainsi positionnée diminue rapidement en fonction de l'éloignement du vecteur d'entrée par rapport au centre de la fonction RBF [SPE 96].

Nous verrons dans le chapitre suivant que ces réseaux de neurones peuvent être utilisés sans passer par la lourde procédure d'apprentissage qui est nécessaire aux perceptrons.

Enfin, il est intéressant de remarquer qu'il existe une équivalence fonctionnelle entre un réseau de neurones RBF et un système d'inférence flou bien que provenant d'origines complètement différentes (les réseaux RBF de la physiologie et l'inférence floue des sciences cognitives) [ROG 93]. Il faut dans ce cas choisir des fonctions d'appartenance gaussiennes de même écart type que celui des RBF ainsi qu'un nombre de neurones égal au nombre de règles d'inférence.

3.4. Applications

Les réseaux de neurones sont utilisés dans de plus en plus de domaines, citons la classification, la reconnaissance de formes, l'identification et la commande de processus. Le choix d'utiliser tel ou tel type de réseau de neurones dépend de l'application mais aussi des capacités de traitement du processeur sur lequel ils s'exécutent. Le caractère local des RBF, par exemple, permet d'alléger les calculs de la sortie du réseau de neurones. En effet, pour une entrée donnée, il n'est pas nécessaire de calculer l'activation de tous les neurones de la couche cachée, mais uniquement ceux dont le domaine de réception couvre l'entrée. Lorsque le nombre de neurones d'un tel réseau est important pour couvrir tout l'espace d'entrée, cette réduction du temps de calcul devient non négligeable.

Il existe évidemment de nombreuses autres variantes de réseaux de neurones [CHE 96], [PIC 94] mais elles ne sont que très partiellement utilisées en commande [MIL 96]. Ce sont des structures moins connues, citons les réseaux de Hopfield ou de Hamming qui ne sont pas du type "propagation directe". Ils nécessitent plus de temps de calcul et leur analyse est moins directe. Les réseaux de Kohonen sont, quant à eux, utilisés principalement en classification.

Dans cette étude, nous nous sommes limités aux réseaux de neurones de type perceptrons et RBF. Ils se prêtent le mieux à notre application. De plus, la relative facilité avec laquelle on peut analyser de leur fonctionnement dans le cadre de la régulation permet de mieux les exploiter.

4. Algorithmes génétiques

4.1. Principe et définitions

Les algorithmes génétiques, comme les réseaux de neurones, font partie des "Réseaux Adaptatifs Non-linéaires" (RAN) [REN 95]. Ils sont composés d'un grand nombre d'unités élémentaires ou agents, qui sont dans notre cas des neurones ou des chromosomes. Ces agents traitent l'information le plus souvent de façon parallèle et distribuée. Ils interagissent entre eux d'une manière non-linéaire et sans contrôle central. Si l'environnement extérieur dans lequel ils baignent est capable de leur fournir une rétroaction, alors les agents et leurs interactions

sont modifiés par des "opérateurs" de telle sorte que le système global s'adapte progressivement à son environnement et améliore sa réponse.

Les algorithmes génétiques sont développés pour des fins d'optimisation. Ils permettent la recherche d'un extremum global. Ces algorithmes s'inspirent des mécanismes de sélection naturelle (Darwin) et de la génétique de l'évolution. Un algorithme génétique fait évoluer une population de gènes en utilisant ces mécanismes. Il utilise une fonction coût basée sur un critère de performance pour calculer une "qualité d'adéquation" (fitness). Les individus les plus "forts" seront à même de se reproduire et auront plus de descendants que les autres. Chaque chromosome est constitué d'un ensemble d'éléments appelés caractéristiques ou gènes. Le but est de trouver la combinaison optimale de ces éléments qui donne un "fitness" maximum. A chaque itération (génération de population), une nouvelle population est créée à partir de la population précédente.

Il existe de nombreuses façons de procéder. Chaque utilisateur conçoit en fait sa variante qu'il juge s'adapter le mieux à son problème. Nous présentons dans ce qui suit, la version finale de l'algorithme génétique que nous avons développé pour notre application.

Tout d'abord, tous les individus sont évalués ; on calcule leur fonction d'adéquation (fitness) et ils sont classés du meilleur au pire (figure 1.20).

Plus l'individu se trouve en haut de la liste, plus il a de chance de se reproduire. Cette phase de reproduction s'effectue en plusieurs étapes de mutation et de croisement.

La mutation consiste à modifier aléatoirement un ou plusieurs gènes d'un chromosome (caractéristiques d'un individu). Alors que le croisement s'effectue en échangeant plusieurs gènes entre deux parents.

Originellement, le codage des individus se faisait en transcrivant en binaire les paramètres à optimiser afin de constituer un gène. Ces gènes sont alors mis bout à bout pour former le chromosome. Il existe cependant une approche appelée codage réel, où les fonctions de mutation et de croisement sont réécrites pour s'appliquer directement au vecteur de paramètres sans passer par la forme binaire. Ces algorithmes se prêtent d'ailleurs fort bien pour donner naissance à des méthodes hybrides qui allient méthodes classiques et algorithmes génétiques [CHO 97]. Nous avons retenu un codage réel, plus souple et plus précis. On évite certains problèmes dus au codage binaire. Le codage réel procure aussi une vision directe des paramètres tout au long de l'évolution de la population.

4.2. Applications

Nous avons utilisé les algorithmes génétiques pour l'identification des paramètres du modèle des machines étudiées ainsi que lors de l'optimisation des régulateurs.

Nous avons travaillé, pour la quasi-totalité des résultats qui seront présentés au chapitre suivant, sur une population de 55 individus.

Les taux de croisement et de mutation sont, non pas issus d'une probabilité qu'a un individu de se croiser ou de se muter, mais plutôt fixés par des tailles de sous-populations dont les fonctions sont déterminées à l'avance.

Le tableau 1.6 résume la configuration retenue.

| Nombre d'individus | Position | Constitution de la sous-population |
|--------------------|----------|---|
| 1 | 0 à 0 | Recopie : on conserve le meilleur individu |
| 9 | 1 à 9 | Mutation au hasard d'individus choisis parmi les 10 premiers avec un facteur au maximum de 0,001 |
| 5 | 10 à 14 | Mutation au hasard d'individus choisis parmi les 15 premiers avec un facteur au maximum de 0,1 |
| 10 | 15 à 24 | Croisement au hasard de deux individus choisis parmi les 15 premiers sans favoriser un parent par rapport à l'autre |
| 10 | 25 à 34 | Mutation au hasard d'individus choisis parmi les 35 premiers avec un facteur au maximum de 0,5 |
| 20 | 35 à 54 | Mutation au hasard d'un seul gène du premier individu de la population avec un facteur au maximum de 0,01 |

Tableau 1.6 Configuration de la population d'individus

La troisième colonne explique l'origine de la sous-population concernée. Par exemple, les individus du rang 10 à 14 sont issus d'une mutation de plus ou moins 10 % à partir d'un des individus de la génération précédente. Ce parent se doit d'être classé parmi les 15 premiers, ce qui signifie qu'on le juge apte à se reproduire.

Chaque population est plus ou moins spécialiste d'une phase du processus d'optimisation. Il apparaît en effet, quand la population part d'une initialisation aléatoire, que ce sont les individus issus d'une mutation avec un fort facteur qui se retrouvent en haut de la liste après évaluation du "fitness" et classement. Quand les individus commencent à tendre vers le vecteur optimum, c'est plutôt la dernière sous-population (faible mutation au hasard d'un seul gène) qui est très sollicitée, en même temps que les individus issus du croisement. Enfin, vers la fin, le meilleur individu reste souvent au sommet de la liste. Ceci peut être retenu comme critère de convergence.

On pourrait croire, à la vue de sous-populations peu actives, qu'il faille réduire leur taille ou les éliminer ; il n'en est rien. En effet, l'un des grands avantages des algorithmes génétiques, c'est de pouvoir sortir des maximums locaux. Des mutations au hasard ainsi que le grand nombre d'individus sont une nécessité si l'on tient à garder cette propriété.

On notera également que l'on a eu à aucun moment besoin de dériver une quelconque fonction. C'est un autre avantage des algorithmes génétiques. Toute médaille ayant son revers, la convergence vers l'optimum demande un grand nombre de générations. Si l'évaluation du "fitness" de chaque individu demande un temps de calcul important, alors le processus devient très lourd. Il se prête cependant très bien au calcul parallèle puisque l'adéquation d'un individu n'est pas liée à celle de son voisin.

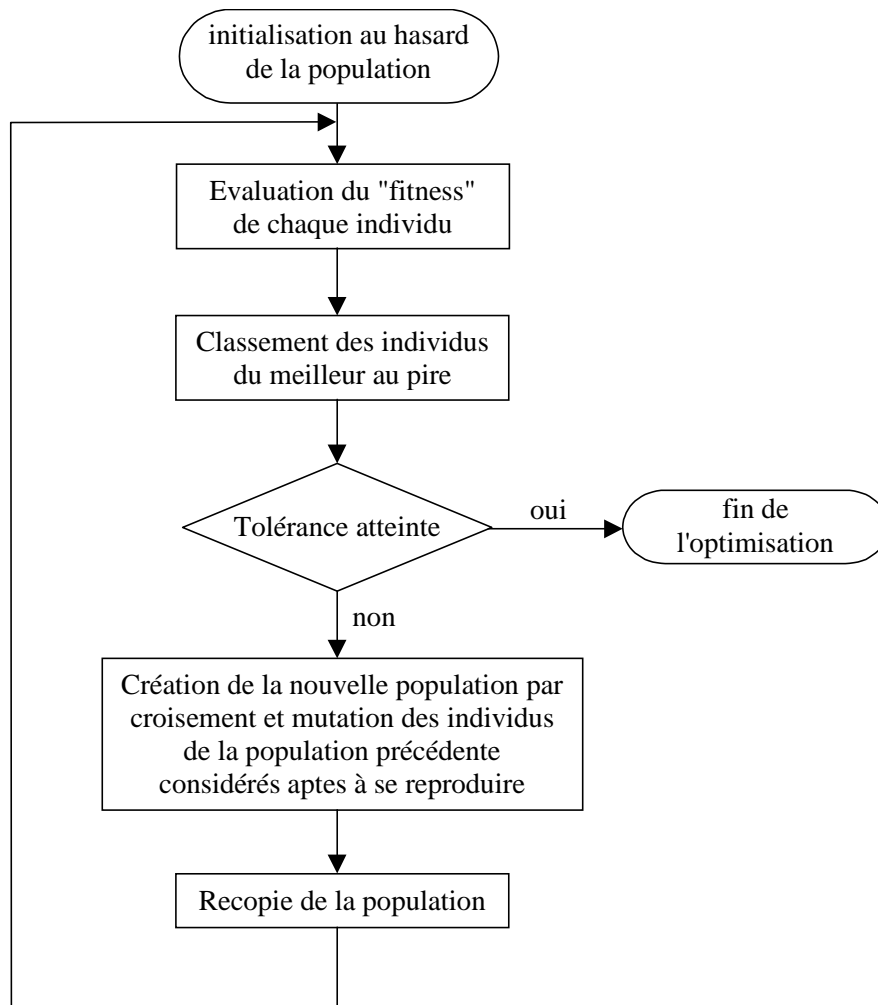


Figure 1.20 Organigramme de l'algorithme génétique

5. Conclusion

Dans ce chapitre, nous avons présenté les bases nécessaires à la compréhension des méthodes à base de logique floue, de réseaux de neurones et d'algorithmes génétiques. Le vocabulaire utilisé par les communautés qui étudient et développent ces méthodes est assez vaste et non encore uniformisé. Il nous a donc semblé nécessaire de préciser celui utilisé ici afin de permettre une lecture claire et sans ambiguïté des chapitres qui vont suivre.

De nombreuses possibilités d'utilisation de ces techniques sont envisageables rien qu'en ce qui concerne la machine asynchrone et sa commande. Nous présenterons les méthodes pour lesquelles nous les avons utilisées et nous nous garderons de faire des conclusions trop hâtives en ce qui concerne leur supériorité par rapport aux méthodes classiques. Ce n'est pas parce qu'une méthode est nouvelle qu'elle est forcément plus efficace. Nous les comparerons donc de manière objective aux méthodes plus classiques. Nous mettrons en lumière ce que ces techniques apportent comme améliorations sans occulter leurs désavantages.