UNIVERSITÉ DU QUÉBEC

RECONNAISSANCE D'ALGUES TOXIQUES PAR VISION ARTIFICIELLE ET RÉSEAU DE NEURONES

MÉMOIRE DE RECHERCHE

PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À RIMOUSKI

Comme exigence
du programme de maîtrise en ingénierie
pour l'obtention du grade de maître ès sciences appliquées (M.Sc.A.)

PAR

RICHARD LEPAGE

09, 2004

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

The author retains ownership of the copyright of this dissertation or thesis. Neither the dissertation or thesis, nor substantial extracts from it, may be printed or otherwise reproduced without the author's permission.

RÉSUMÉ

Depuis de nombreuses années, on cherche à identifier rapidement si des zones de pêches côtières ont été contaminées par des algues toxiques. Une volonté affirmée se développe quant à la conception d'un système automatisé de reconnaissance d'organismes cellulaires à traitement différé ou en temps réel; l'utilité d'un tel système étant avant tout de palier les délais d'analyse taxonomique qui prennent plusieurs jours à s'effectuer avant d'en arriver à une conclusion définitive. De ce fait, la prévention est tardive et les risques plus élevés de contamination toxique pour la population humaine et animal (huîtres, moules, poissons...). La situation, en ce qui a trait aux systèmes existants pour la reconnaissance automatisée du phytoplancton, est en plein développement. Il y a néanmoins des difficultés inhérentes quant à la précision de la classification et de la détection.

L'objectif de ce projet de recherche est la reconnaissance automatisée d'une espèce spécifique de phytoplancton, soit l'*Alexandrium tamarense*. Un appareil, que l'on nomme FLOWCAM, fournit des données (images et informations cytométriques) provenant d'échantillons prélevés à des sites spécifiques. Ces données permettent à un système de reconnaissance de discriminer les espèces végétales contenant des toxines de celles qui sont inoffensives pour l'espèce humaine.

On privilégie deux stratégies dans ce mémoire de recherche, la première est une fusion de données obtenues par traitement d'images associées à des paramètres cytométriques générés par le FLOWCAM et la seconde est d'utiliser la totalité des paramètres cytométriques sans procéder au calcul des moments d'ordre $n<3$ sur chacune des images. Pour les deux approches, la discrimination des classes est effectuée par un réseau de neurones dont l'optimisation a été calculée par la méthode des plans d'expériences de Taguchi.

Dans les deux stratégies, nous avons émis l'hypothèse que les données présentaient une distribution normale. À partir de cette distribution, 33% des données provenant de la base de données #1 et #2 ont été choisies de manière aléatoire pour être présenté au réseau de neurones lors de l'étape d'apprentissage. Nous avons ensuite présenté les données provenant de la base de données #1 et #3 au système de reconnaissance pour les deux stratégies.

Les résultats que nous avons obtenus permettent de conclure que le calcul des paramètres par la méthode des moments d'ordre $n<3$ dans la stratégie #1 est imprécise et inadéquate comparativement à la stratégie #2 dont la discrimination se base uniquement sur les paramètres existants du FLOWCAM. Une étude plus exhaustive sera donc nécessaire pour vérifier si la méthode des moments pour $n>3$ fournit des résultats plus probants.

AVANT-PROPOS

Ce travail a été effectué au sein du laboratoire de recherche en génie électrique à l'Université du Québec à Rimouski, dirigé par le professeur Jean-François Méthot.

Je désire remercier mon directeur de recherche, M. Jean-François Méthot, pour la qualité de son aide, sa disponibilité ainsi que sa patience et son amabilité durant la première correction de ce mémoire. Je désire également remercier mon co-directeurs, M. Jean-François Dumais, pour le soutien moral qu'il m'a apporté durant mes études à la maîtrise et sa confiance envers mes capacités à poursuivre mes études. Je tiens à remercier en particulier M. Maurice Levasseur, du département de biologie de l'Université Laval du Québec qui a eu l'amabilité de me fournir les données permettant de tester adéquatement le système de reconnaissance dans son ensemble.

Je tiens à remercier également M. Abderrazak El Ouafi qui présidait le jury d'évaluation ainsi que M. Pierre-Martin Tardif qui évalua le mémoire de recherche en tant qu'évaluateur externe de l'Université du Québec à Rimouski.

Je remercie également mes coéquipiers de travail qui partageaient le même laboratoire et avec lesquels j'ai eu des discussions des plus intéressantes et fructueuses sur nombre de sujets.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ACRONYMES

Acronymes	Définition
FCM	Fichier de données produit par le FLOWCAM contenant des paramètres calculés sur les images et des paramètres cytométriques.
FFT	Transformée de Fourier rapide (Fast Fourier Transform).
FPGA	Field-Programmable Gate Array (Composant électronique à logique programmable).
IFREMER	Institut français de recherche pour l'exploitation de la mer
MLP	Multi Layer Perceptron (Perceptron multi couches)
PCA	Principal Component Analysis (Analyse par composantes principales)
RBF	Radial Basis function (Fonctions de base radiale)
RNA	Réseau de neurones artificiels
SOM	Réseau de neurones à connections totales dont le meilleur représentant est le réseau de Kohonen (Self Organised Map).

INTRODUCTION

Le sujet de ce mémoire de recherche est la reconnaissance automatisée d'une espèce spécifique de phytoplancton ou d'algue transportant des toxines qui sont absorbées par des produits de la mer à consommation humaine. On effectue cette reconnaissance automatisée par le biais de techniques appropriées que l'on retrouve tant en vision artificielle, en statistique et en traitement de signal. L'utilisation des réseaux de neurones est une approche que nous avons considérée dans cette recherche. Lors de l'expérimentation, l'analyse fut effectuée sur une espèce particulière (*Alexandrium tamarense*). On retrouve celle-ci dans l'estuaire du Saint-Laurent et elle est à l'étude depuis quelques années par plusieurs instituts des sciences de la mer dont l'institut Maurice-Lamontagne. Pour tester le système de reconnaissance, on dispose de données provenant d'un appareil (le FLOWCAM) qui permet d'obtenir des images ainsi que des paramètres cytométriques obtenus à partir d'échantillons cellulaires, ceux-ci circulant individuellement dans une veine transparente. Le FLOWCAM est la combinaison d'un microscope de grossissement 20X à 40X et d'un cytomètre à flux (laser à fluorescence).

Le cytomètre du FLOWCAM est constitué de un ou plusieurs lasers de longueur d'onde définie qui excitent le pigment de la cellule circulant dans la veine transparente et dont les photons diffusés sont captés par des détecteurs de longueurs d'onde spécifiques. Ces détecteurs sont des photos-multiplicateurs ou amplificateurs de photons.

L'ensemble des détecteurs donne des informations sur l'épaisseur, la longueur, le nombre de noyaux cellulaires ainsi que d'autres paramètres que contient l'échantillon à analyser. Lorsque les détecteurs captent un signal, un circuit de contrôle engage simultanément le déclenchement d'une caméra combinée à un microscope permettant de saisir l'image de la cellule. Cette image donne ensuite une information supplémentaire quant à la couleur, la texture et la forme complète de l'échantillon. Le FLOWCAM **[5]**, est l'un des appareils de dernière génération permettant un classement automatique par groupement de cellules. Les cellules individuelles de phytoplancton ne sont pas reconnues comme telles; il s'agit beaucoup plus d'un dénombrement grossier par classes de caractéristiques. Parmi ces caractéristiques nous notons la dimension de la cellule, le type de pigment de fluorescence (phycoérythrine, chlorophyle) ainsi que d'autres. Nous verrons au chapitre un (1) que ce classement par regroupement de caractéristiques est insuffisant pour détecter les cellules toxiques afin de prédire l'arrivée prochaine d'un bloom (prolifération rapide d'une algue).

Dans ce document, nous expérimentons et discutons dc la possibilité de classifier automatiquement une cellule distincte par rapport à d'autres espèces apparentées en combinant les données fournies par le FLOWCAM qui se présentent sous forme d'images et de paramètres cytométriques. Les données obtenues par le FLOWCAM sont traitées au préalable pour en extraire des caractéristiques qui seront par la suite présentées à un réseau de neurones pour l'étape finale de la classification.

Un des problèmes régulièrement rencontrés, lorsque l'on traite des images, est la difficulté d'obtention de caractéristiques invariantes sous translation, rotation ou homothétie. La technique expérimentée dans ce mémoire pour extraire des caractéristiques invariantes est le calcul des moments d'ordre n. Cette méthode de détermination des paramètres invariants est due à sa fiabilité et non pour sa rapidité en temps de calcul dans le cadre de cette expérimentation. On risque de rencontrer des problèmes au niveau du temps de calcul si on vise à considérer le traitement en temps réel.

Tout au long de cet ouvrage, la simulation a été effectuée par l'intermédiaire du logiciel MATLAB (version 6.5.0.180913a). Ce logiciel dispose de plusieurs librairies telle que la librairie « **Neural Networks** » permettant de concevoir des réseaux de neurones à différentes configurations, la librairie « **Image Processing** » pour le traitement des images, la librairie « **Wavelet** » pour le traitement des signaux ainsi que la librairie « **Statistics** » pour la normalisation des données et l'analyse par composantes principales.

Des tests expérimentaux ont été effectués pour l'obtention de la configuration optimale du réseau de neurones en se servant de la méthode des plans d'expériences selon Taguchi **[30,31]** ; la théorie pour configurer convenablement un réseau de neurones de manière optimale (nombre de couches et nombre de neurones) faisant défaut jusqu'à ce jour.

Dans la littérature, il est fait mention du fait que certaines expériences portant sur la classification du phytoplancton au moyen de la vision artificielle étaient vouées à l'échec dû à la complexité des formes et aux espèces apparentées apparaissant dans le même échantillon **[6,7,23];** les espèces apparentées, non toxiques, pouvant être une source de nombreuses fausses alertes. Divers travaux ont démontré que l'utilisation de paramètres cytométriques permettrait une très bonne discrimination sur des appareils autres que le FLOWCAM **[43]**.

L'apport nouveau du projet de recherche est de vérifier la possibilité de fusionner les deux types de données obtenus par le FLOWCAM, soit l'image de la cellule en association avec les informations cytométriques de celle-ci. Par un traitement adéquat de l'image, on arriverait à différentier les cellules transportant des toxines de celles qui sont inoffensives pour l'espèce humaine.

Le logiciel prototype pourrait servir de base technologique pour la conception d'un système de détection et de reconnaissance en temps réel. Celui-ci pourrait éventuellement être réalisé sous forme de logique câblée (FPGA) et pourrait être installé de manière permanente sur des bouées proches de zones côtières. Un signal transmis par ondes radio à une base terrestre avertirait de la présence d'algues toxiques dans le secteur sous étude, les autorités ayant alors le temps de réagir pour avertir la population sur l'emplacement des zones contaminées.

On pourra éventuellement utiliser ce système de reconnaissance pour l'identification d'autres espèces de phytoplancton et, qui sait, ce dernier permettra éventuellement de découvrir de nouvelles espèces. Le logiciel a été écrit en code MATLAB pour une meilleure reproductibilité et une facilité dans la compréhension du code.

Ce mémoire se subdivise en plusieurs étapes. Nous avons porté une attention particulière sur les outils mathématiques utilisés lors de l'expérimentation. Nous avons également donné quelques exemples permettant de bien saisir les concepts mis en jeu dans cette recherche avant de les appliquer à l'expérimentation proprement dite.

Le chapitre un fait référence à la problématique du travail de recherche ainsi qu'aux difficultés que d'autres chercheurs et chercheuses ont rencontré dans la reconnaissance automatique du phytoplancton. À partir de ces difficultés, des hypothèses ont été échafaudées permettant de solutionner adéquatement le problème. Les hypothèses qui sont utilisées dans ce travail de recherche font l'objet d'une discussion pour arriver à une solution sur d'autres problèmes du même type.

Les chapitres deux, trois et quatre sont dédiés aux techniques utilisées en reconnaissance de formes, aux méthodes d'extraction de caractéristiques de formes ou de paramètres, aux traitements effectués sur les images et les signaux ainsi qu'à la classification de données par réseaux de neurones. Ces chapitres seront appréciés par les néophytes, les spécialistes pouvant se diriger vers les chapitres cinq (5) et six (6) qui représentent le cœur de ce mémoire.

Le chapitre deux fait un rapide survol de différentes techniques de prétraitement des images et des signaux. Ce processus est d'une grande importance pour normaliser les données d'entrée et de sortie ainsi que pour déterminer des caractéristiques invariantes en translation, en rotation et en homothétie. Le choix d'une technique de prétraitement des données est une étape préliminaire à l'extraction des paramètres. Après cette étape préliminaire, l'étape de classification s'en trouve facilitée fournissant de par ce fait la possibilité d'un traitement plus rapide lors d'une utilisation en temps réel.

Le chapitre trois présente des techniques d'extraction de paramètres, une étape essentielle avant le processus de classification. L'extraction des paramètres permet d'éliminer les données ayant une trop forte corrélation entre elles; c'est en quelque sorte une technique de compression des données. La technique utilisée dans ce document est la méthode PCA (Analyse par composantes principales) que nous verrons en détail.

Dans le chapitre quatre, on examine et discute des techniques existantes en classification et on élabore sur les meilleures méthodes de classification qui se retrouvent dans l'industrie. Le réseau de type MLP (Multi layer Perceptron) est l'outil de classification que nous utiliserons en association avec la méthode de calcul des poids par rétropropagation du gradient de l'erreur. Les algorithmes de convergence que nous expérimenterons dans cette recherche seront l'approche de Levenberg-Marquard (matrice hessienne approximée) ainsi que l'approche BFGS (Broyden, Fletcher, Goldfarb, et Shanno), un algorithme de type quasi-Newton **[44]**.

Le principe de fonctionnement du réseau à couches multiples (MLP) sera détaillé de même que la méthode d'apprentissage par rétropropagation de l'erreur du gradient que l'on appelle communément « back propagation ». Nous verrons quelques techniques de convergence qui sont plus ou moins rapides tout en offrant des caractéristiques intéressantes.

Ces techniques de convergence prennent comme référence l'algorithme « back-propagation ». Nous parlerons également de la méthode statistique bayésienne que l'on retrouve entre autre dans les réseaux RBF (Radial Basis Function).

Le chapitre cinq décrit la méthodologie employée lors de la phase expérimentale pour concevoir une simulation fonctionnelle du logiciel de détection et de classification d'algues toxiques. La démarche expérimentale est une étape importante pour la reproductibilité des résultats par les pairs. À partir de cette démarche, on peut généralement découvrir les erreurs qui se sont produites et dégager les mesures nécessaires pour un travail de recherche plus complet.

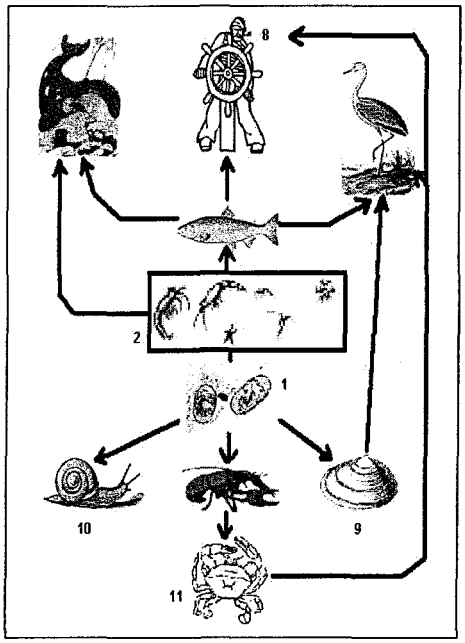
Finalement, le chapitre six contient la discussion des résultats obtenus et la conclusion de ce travail de recherche. On y discute également des options qui amélioreraient la qualité de la classification ainsi que la vitesse de traitement. On y discute également des possibilités d'application de ce travail de recherche dans d'autres secteurs d'activité.

Il est entendu que cette recherche ne se veut pas exhaustive, son objectif étant de déterminer si la fusion des paramètres de forme associés à des paramètres cytométriques permet de discriminer une espèces spécifiques de phytoplancton par rapport à des espèces apparentées. Cette fusion des données paramétriques apportera certainement des éléments de réponse à cette voie de recherche. Nous avons inclus dans la section théorique des informations supplémentaires concernant le traitement des signaux, l'extraction de paramètres par la méthode des ondelettes ainsi que la reconnaissance de signaux par un réseau de fonctions de base radiale qui est particulièrement bien adapté pour la reconnaissance de signaux analogiques.

CHAPITRE 1

PROBLÉMATIQUE

Le phytoplancton est le principal aliment des coquillages filtreurs. Il est indispensable à leur croissance et contribue à leur engraissement. Mais sur les milliers d'espèces qui constituent cette nourriture, quelques unes sont indésirables et responsables de toxicité des coquillages; c'est le cas notamment de deux espèces, soit *Alexandrium tamarense* et *Pseudo-nitzchia multiséries*. On peut voir sur la figure 1.1 le cycle de contamination des produits de la mer à consommation humaine.



Cause : accumulation de phosphates (augmente les nutriments)

Effet : Prolifération de phytoplancton.

1 Bloom d'algues toxiques (Prolifération rapide de phytoplancton)

2 Zooplancton herbivore

8 Consommation par l'être humain de bivalves (plus à risque)

9 Bivalves (moules, huîtres, palourdes,...)

10 Les coquillages fouisseurs

(accumulation des toxines après le bloom)

11 Crabes, nécrophages,...

Seules certaines espèces de phytoplancton contiennent des toxines

Informations provenant de IFREMER (voir glossaire)

Figure 1.1 Cycle de consommation d'algues toxiques par les mollusques

Comme pour les végétaux terrestres, le phytoplancton a besoin, pour se développer, de lumière, de température, de sels nutritifs, de vitamines,... et d'espace. Ainsi, comme ils sont nombreux à posséder les mêmes besoins, les espèces développent des aptitudes à lutter les unes contre les autres pour occuper un secteur et pour lutter contre le zooplancton (plancton animal) qui constitue leur premier prédateur. La dimension d'une cellule végétale que l'on nomme également phytoplancton se situe entre 1 et 200 micromètres. Pour donner un ordre de grandeur, un phytoplancton dont la dimension est de 200 micromètres est visible sans l'aide d'un quelconque instrument d'optique. En comparaison, l'épaisseur d'un cheveu est d'environ 80 micromètres.

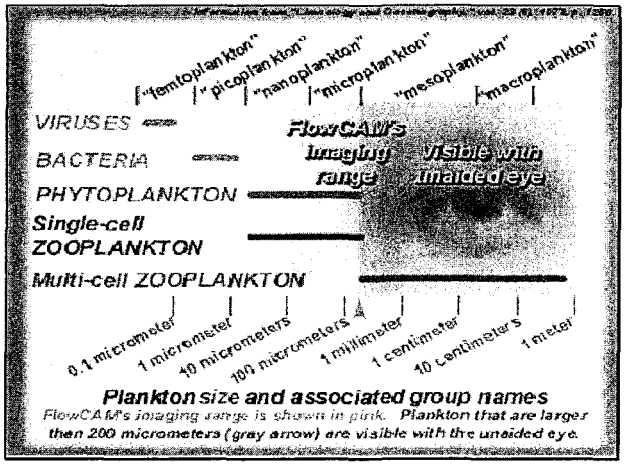


Image provenant du site http://www.bigelow.org/flowcam/

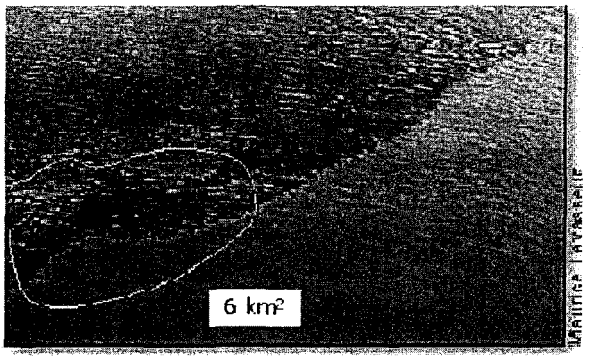
Figure 1.2 Ordres de grandeur du phytoplancton

La figure 1.2 permet de se faire une idée de l'ordre de grandeur du phytoplancton. Le FLOWCAM permet l'analyse du phytoplancton se situant entre 1 micromètre et 1 millimètre.

Certaines espèces, comme *Alexandrium tamarense,* produisent des toxines. Les coquillages filtreurs, en se nourrissant, absorbent ce phytoplancton et concentrent, après digestion, ces toxines dans leur chair. Si d'une manière générale une toxine n'est pas dangereuse pour les coquillages, il n'en va pas de même pour les êtres humains qui les consomment. Selon un rapport de pêches et océans Canada sur le monitorage du phytoplancton toxique dans le Saint-Laurent entre 1989 et 1994 **[1]**, les principales algues nuisibles répertoriées sont : *Alexandrium tamarense (Gonyaulax tamarensis), A. ostenfeldii, Dynophysis acuminata, D. norvegica* et *Phalacroma rotundatum*. La diatomée *Pseudo-nitzchia multiséries* n'a toutefois jamais été identifiée aux stations de monitorage située dans l'estuaire et le golfe du Saint-Laurent. Dépendant des saisons de l'année, on retrouve *Alexandrium tamarense* durant les saisons plus chaudes tandis que *Pseudo-nitzchia multiséries* prolifère habituellement durant les saisons froides.

On détecte généralement la présence d'algues toxiques en grande quantité par la formation d'une nappe flottant à la surface de l'eau ayant une couleur et une odeur caractéristiques. La formation de cette nappe est très rapide et habituellement les toxines ont été absorbées par les mollusques depuis plusieurs jours. Cette prolifération rapide de phytoplancton porte le nom de « bloom ». Sur la figure 1.3, nous montrons justement ce que représente un « bloom » par une nappe visible plus foncée qui peut s'étendre très rapidement et couvrir une très grande superficie.

Le « bloom » ou fleurs d'eau se produit suite à une combinaison d'ensoleillement, d'élévation de la température et d'augmentation des nutriments (azote, phosphore, parfois de la silice). Il entraîne une efflorescence anormale de diatomées ou de dinoflagellés, qui en général se produit le printemps avec une récidive estivale ou automnale si un renouvellement des nutriments se produit.



Courtoisie de M. Maurice Levasseur, Département de Biologie Université Laval

Figure 1.3 Bloom d'algues toxiques (phytoplancton)

À l'occasion, les concentrations d'algues toxiques deviennent assez élevées pour donner une coloration rougeâtre à l'eau. En août 1996, une telle «marée rouge» s'était produite dans le Saint-Laurent. Il est probable que cette floraison trouve sa stimulation dans les pluies diluviennes et les forts vents qui ont causé des inondations dévastatrices sur la rive-nord du Saint-Laurent et au Saguenay pendant cette période. Cette marée rouge a occasionné des mortalités de lançons et de goélands le long de la côte-nord de la péninsule gaspésienne.

Les concentrations qui s'avèrent dangereuses pour la consommation humaine sont de l'ordre de **5 à 50 cellules par ml** pour *Alexandrium tamarense* et de **100 cellules et plus par ml** pour *Pseudo-nitzschia multiséries* **[1].** Pour prévoir la formation d'un « bloom » une semaine à l'avance, il faut être en mesure de pouvoir détecter une cellule par ml **[2]** ce qui est considéré comme une très faible concentration et donc assez difficile à détecter. Le FLOWCAM permet une détection de cette précision mais sans être en mesure de la détecter de manière automatique.

Le choix dans la détection de l'espèce *Alexandrium tamarense* vient du fait que sa toxine est mortelle pour l'être humain et qu'elle prolifère plus souvent que l'espèce *Pseudo-nitzschia multiséries*. On associe à *Alexandrium tamarense* la toxine PSP (*Paralytic Shellfish Poisoning*) et à *Pseudo-nitzschia multiséries* la toxine ASP (*Amnesic Shellfish Poisoning*). On retrouve d'autres algues dont les toxines entraînent des désagréments mais aucune mortalité. Ces toxines inoffensives quoiqu'incommodantes sont les toxines DSP (*Diarrhetic Shellfish Poisoning)* et NSP (*Neurotoxic Shellfish Poisoning*).

Pour les algues présentant une toxicité moindre, les symptômes de **gastro-entérites** peuvent se manifester dès quatre heures après la consommation et perdurer jusqu'à quatre jours; nausées et vomissements, diarrhées, douleurs intestinales parfois doublées de frissons sont les effets causés par ces toxines, soit une forte gastro-entérite, principalement provoquée par les toxines émises par l'algue *Dinophysis.* Le tableau 1.1. est un résumé des observations effectuées sur les algues toxiques.

De manière générale, les endroits où des cas de mortalité ont été répertoriés se situent en Alaska et dans certaines régions du Canada dans le cas de *Pseudo-nitzschia multiséries* tandis qu'*Alexandrium tamarense* a été retrouvé aux État-Unis. On en retrouve également au Canada **[1]**.

Tableau 1-1 Algues toxiques et symptômes

Phytoplancton	Toxine	Sites observés	Symptômes
Alexandrium	PSP Paralytic Shellfish Poisoning	Nouvelle Angleterre; Côte Ouest (incluant l'Alaska)	Perte de coordination. Difficulté respiratoire dans les cas sévères. Cette toxine peut être fatale.
Pseudo-nitzschia	ASP Amnesic Shellfish Poisoning	Aucune cause de maladie humaine dans les U.S.A.. On en retrouve par contre au Canada.	Crampes abdominales, désorientation. Perte permanente de la mémoire dans les cas sévères. Peut être fatale.
Gymnodium breve	NSP Neurotoxic Shellfish Poisoning	Côte sud-ouest, Golf du Mexique	Gastro-entérite. Amplification douloureuse des sensations. Aucune mortalité
Dinophysis	DSP Diarrhetic Shellfish Poisoning	Aucune cause de maladies rapportés dans les U.S.A.	Gastro-entérite. Non fatale.

Données provenant du site de l'Université du Maryland http://aquaticpath.umd.edu/toxalg/

Face à cette nouvelle manifestation de la pollution des eaux par les nitrates et le phosphore, divers moyens de détection ont été utilisés mais aucun ne s'avère assez rapide, le « bloom » ayant déjà fait son œuvre dans la contamination des moules et des huîtres.

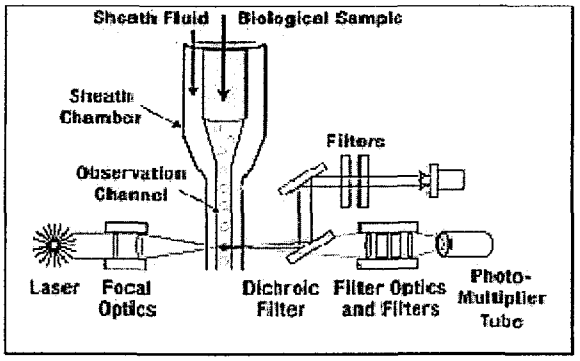
La mise en œuvre d'une méthode de détection sensible permettant d'identifier automatiquement une cellule par ml de l'espèce toxique permettrait de prévoir la production d'un « bloom » dès le début et non après quelques jours.

Actuellement, il faut de deux à trois jours **[2]** pour diagnostiquer la présence de micro-organismes pathogènes (bactéries, amibes, algues toxiques) dans l'eau, ce délai étant dû à l'étape de culture. Une analyse taxonomique superficielle est effectuée, prenant de deux à trois jours et on ajoute également à ce délai l'étape de prélèvement qui prend de un à deux jours, le tout prenant typiquement une semaine et plus.

La détection rapide et sensible des algues toxiques représente un enjeu majeur dans le domaine sanitaire compte tenu des accidents alimentaires qui se sont manifestés ces dernières années à la suite de consommation de coquillages contaminés. Bien que certaines algues toxiques fassent l'objet d'un programme de surveillance permanente, elles ne sont le plus souvent identifiées que trop tardivement; par exemple, à l'occasion d'une phase de prolifération rapide.

Pour être en mesure de détecter la présence d'algues toxiques avant prolifération, il faut disposer d'une technique de détection qui soit très sensible pour détecter une très faible concentration. Il existe actuellement une technique de détection de micro-organismes pathogènes s'apparentant à la cytométrie **[2,5,6]**. Les micro-organismes sont repérés et comptés par cytométrie en phase solide. Sur la figure 1.4, on présente le schéma simplifié de ce type de cytomètre.

La cytométrie fonctionne selon le type de fluorescence retrouvée dans une cellule végétale. Dépendant de l'intensité de la fluorescence, on distingue les cellules organiques des particules inorganiques en suspension dans le jet de circulation. On remarquera également que les cellules sont en suspension dans une burette interne à un cylindre contenant un fluide circulant à grande vitesse. Cette vitesse est toutefois limitée pour des raisons d'intégralité cellulaire, i.e qu'il ne faut pas que la cellule subisse des dommages au niveau de sa structure physique.



Courtoisie de l'Université du Michigan (Collège of Engineering)

Figure 1.4 Schéma d'un cytomètre à écoulement fluidique

La figure 1.5 schématise un appareil du même type que le FLOWCAM. Ce schéma plus détaillé donne des informations concernant les longueurs d'onde des lasers utilisés pour activer la fluorescence des cellules organiques. Le FLOWCAM est un appareil qui est en mesure de rencontrer la sensibilité requise pour prévoir un « bloom » une semaine à l'avance.

Le FLOWCAM utilise une combinaison microscope et laser permettant de donner plus d'information et ainsi fournir la possibilité d'une meilleure discrimination pour la distinction des cellules dans un groupe donné. Les données auxquelles font référence cette expérimentation proviennent du prototype que détient le département de biologie de l'Université Laval.

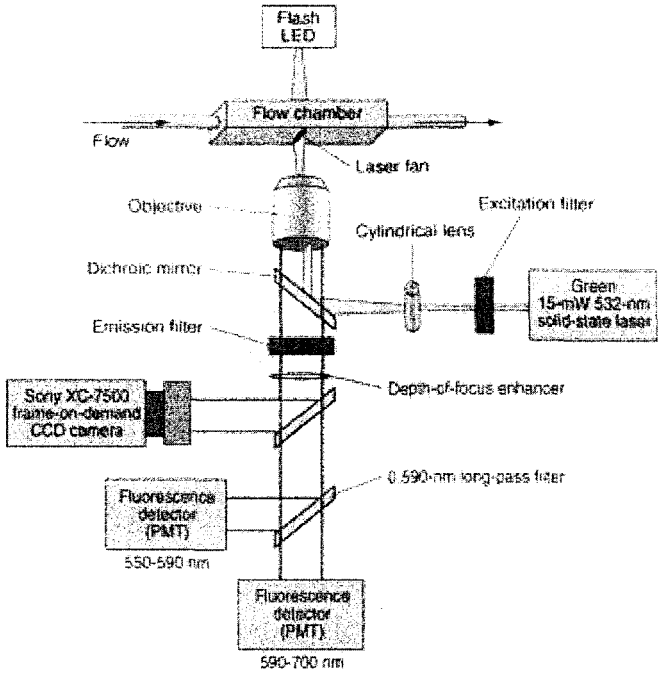
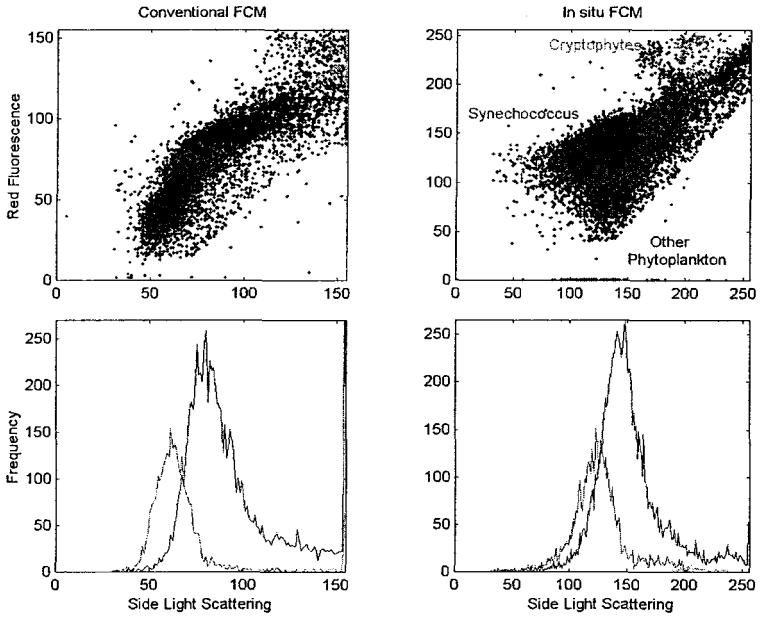


Image fourni par VisionSystem Design

Figure 1.5 Combinaison microscope et laser du FlowCam.

Les données cytomètriques sont visualisées sous forme de nuages de points qui regroupent des organismes selon des caractéristiques de taille et de fluorescence. Par la nature de l'appareil, des espèces distinctes se confondent dans ces groupements.

Sur la figure 1.6, on retrouve un exemple de nuages de points contenant trois couleurs, chaque couleur étant associée à un ensemble de caractéristiques distinctes.



http://www.bigelow.org/flowcam/flo_r1.

Figure 1.6 Représentation par nuages de points d'organismes multiples

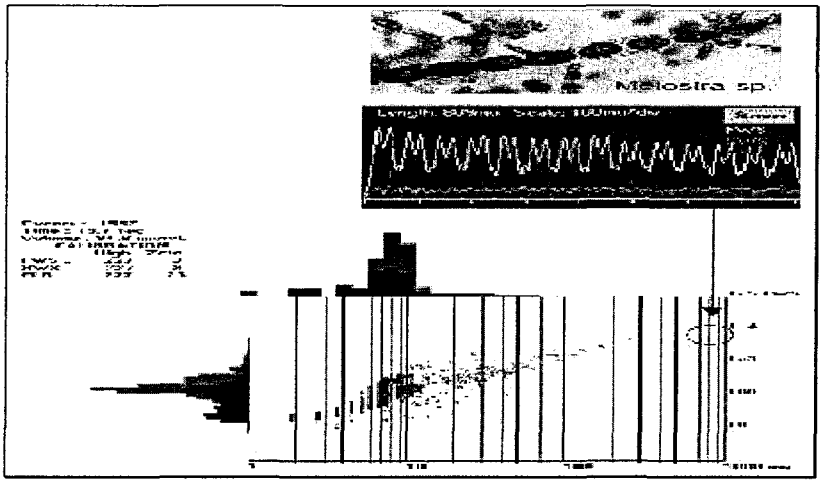
À ce jour, la seule technique appropriée pour une détection à faible concentration [**5**] est un laser argon qui balaye la surface totale de la membrane filtrante. La viabilité des cellules étant fondée sur l'activité métabolique endogène et l'intégrité de leur membrane, les micro-organismes en état de fonctionnement apparaissent fluorescents après clivage par une estérase (classe d'enzyme provoquant la coupure d'une molécule avec fixation d'eau) du substrat fluorogénique qui sert de révélateur. En matière de sécurité alimentaire, un repérage rapide d'une très faible quantité de micro-organismes pathogènes est un véritable défi.

L'appareil qui fournit les données faisant partie de cette expérimentation s'approche de la sensibilité requise pour détecter des cellules à des concentrations aussi faibles qu'une cellule par millilitre. Actuellement, les techniques de détection par cytométrie ne permettent que d'identifier des groupes de cellules au lieu d'identifier des espèces de manière distincte **[5]**, chaque groupe comprenant des dizaines d'espèces; les espèces toxiques, se fondant dans un groupe, échappent alors à toute détection.

Un choix judicieux des paramètres pourrait permettre une meilleure classification des cellules en rapport avec les images obtenues par le FLOWCAM. Nous verrons que le choix des paramètres à partir du calcul des moments d'ordre « n » est une étape importante dans la qualité de la classification en association avec les paramètres cytométriques rattachés à la fluorescence des cellules.

Dans les travaux antérieurs, les paramètres cytométriques étaient inexistants et on essayait à partir de la forme de la cellule d'identifier une cellule végétale spécifique par rapport à d'autres cellules apparentées sans succès appréciable. Dans ce travail de recherche, nous vérifions si la fusion traitement d'image et paramètres cytométriques est une technique viable que nous pouvons suivre pour la détection rapide d'algue toxique à partir des données fournies par le FLOWCAM.

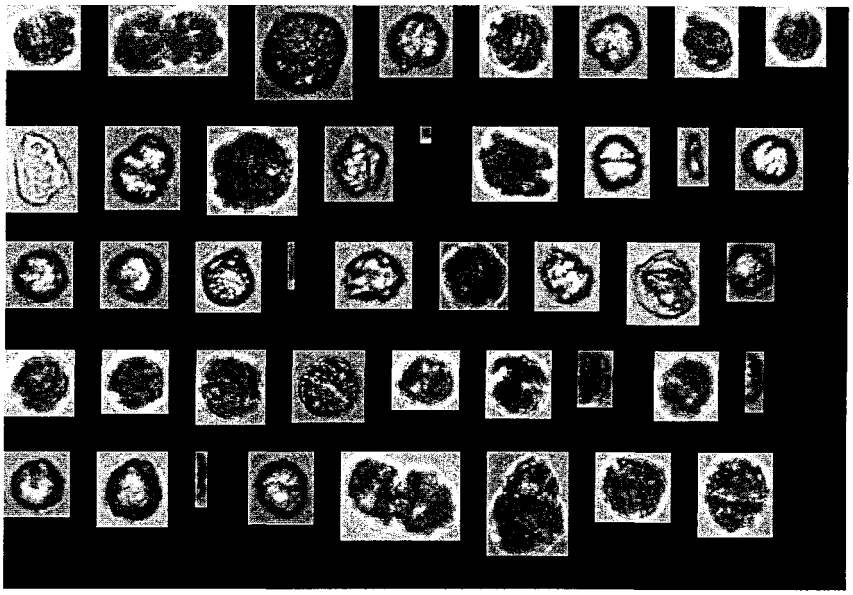
On peut voir à la figure 1.7 un exemple d'image et de signaux que le FLOWCAM peut produire durant une expérience typique. On peut remarquer que l'image au-dessus de la bande contenant le signal de l'échantillon analysé est celle émise par le système d'imagerie du microscope.



http://www.bigelow.org/flowcam/flo_r1.html

Figure 1.7 Image et signaux fourni par le FLOWCAM

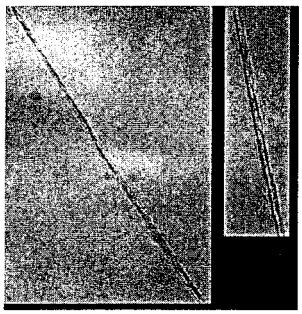
Les figures 1.8 et 1.9 sont des images fournies par le FLOWCAM et donnent une vue détaillée de deux espèces de phytoplancton dont l'une est celle que nous analysons dans cette recherche. Le microscope ayant servi à produire ces images avait un grossissement compris entre 20x et 40x. Comme on peut le voir pour *Alexandrium tamarense*, la forme est très simple quoiqu'il y ait des espèces qui soient très apparentées et qui ne présentent pas la toxicité d'*Alexandrium tamarense*. À partir de l'image et des paramètres cytométriques, nous verrons s'il y a possibilité de discriminer les espèces apparentées des espèces toxiques.



http://www.bigelow.org/flowcam/flo_r1.html

Figure 1.8 Exemples d'Alexandrium Tamarense

L'espèce *Pseudo-nitzschia multisérie* qui n'est pas à l'étude dans cette recherche appartient à la famille des pluricellulaires et peut-être plus difficile à différencier par rapport à d'autres qui présentent des similitudes au niveau de la forme en général; car plus nombreuses encore sont les espèces pluricellulaires. La figure 1.9 présente la forme de cette cellule.



http://www.bigelow.org/flowcam/flo_r1.html

Figure 1.9 Pseudo-nitzschia

Le FLOWCAM produit des paramètres caractéristiques pour chaque cellule passant devant ses capteurs mais pas en nombre suffisant pour une discrimination fiable. En plus de l'image, on pourrait disposer d'une combinaison de signaux provenant du cytomètre. Pour l'instant, l'appareil ne donne que la valeur maximale et le temps de passage de la cellule (largeur de l'impulsion). La forme complète des signaux, si elle était disponible, serait une information très pertinente pour la reconnaissance.

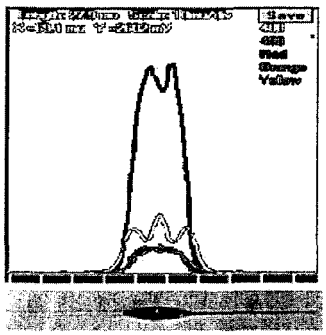


Figure 1.10 Signaux cytométrique d'Alexandrium tamarense provenant du FLOWCAM.

Sur la figure 1.10, on peut voir un exemple d'une telle forme d'onde. Celle-ci est composée de trois signaux provenant de trois capteurs détectant les intensités de fluorescence pour trois longueurs d'onde spécifiques. Ces signaux pourraient servir à mieux caractériser les cellules à reconnaître par le choix et l'extraction de paramètres variés en fonction de la forme des signaux. Dans la figure 1.10, les signaux sont associés à l'espèce *Alexandrium tamarense*.

Au cas où ces signaux deviendraient disponibles dans un futur proche, nous avons ajouté dans ce mémoire une partie théorique sur le traitement des signaux par ondelettes (section 2.2.4.2) qui permet de déterminer les caractéristiques d'un tel signal.

La dernière étape du traitement est la classification par un réseau de neurones de type RBF ou de type MLP dont les entrées sont les paramètres choisis, extraits et calculés au préalable. Des paramètres issus de la forme des signaux d'absorption permettraient de distinguer des signaux entres eux. Nous avons également ajouté une section dans ce mémoire concernant l'apprentissage de signaux par un réseau de neurones de type MLP ainsi qu'un exemple de code MATLAB en annexe.

CHAPITRE 2

LA RECONNAISSANCE DE FORMES

2.1 Introduction

Dans ce chapitre, nous discutons de diverses méthodes de prétraitement des données qui se présentent avant la phase d'extraction des paramètres. Cette phase est relative à la suppression du bruit, à la correction des erreurs, à l'homogénéisation, à la normalisation ainsi qu'à la réduction des données. Parmi les méthodes proposées, quelques-unes seront considérées dans la partie expérimentale de ce mémoire. Il est clair que meilleur sera le prétraitement, plus simple sera la configuration des réseaux de neurones et plus précise sera la classification des données présentées à l'entrée.

Après avoir parcouru l'ensemble des méthodes de prétraitement des données, nous expliquerons pourquoi nous utilisons une méthode plutôt qu'une autre dans le cadre de cette recherche. Nous donnerons également dans cette partie les fonctions que nous utiliserons dans l'environnement MATLAB. Nous utiliserons certaines de ses fonctions pour la conception du programme de reconnaissance du phytoplancton à l'étude. Ce programme pourra servir à un projet plus vaste servant à détecter et classifier différentes espèces de phytoplancton en temps réel, ce qui serait une première en taxonomie automatisée, en prévention écologique et en prévention d'intoxication alimentaire.

2.2 Processus généraux de reconnaissance de forme

L'approche présentée dans ce document permettra au lecteur de s'initier rapidement au domaine de reconnaissance des formes et le préparera à la lecture de documents plus spécialisés. Selon une approche considérée aujourd'hui comme classique, on retrouve principalement six (6) étapes dans l'élaboration du processus complet de reconnaissance de formes [4], soit :

- l'étape du monde physique
- l'étape du codage
- l'étape de prétraitement
- l'étape de l'analyse
- l'étape de l'apprentissage
- l'étape de décision.

Ces étapes sont visualisées sur la figure 2.1 .

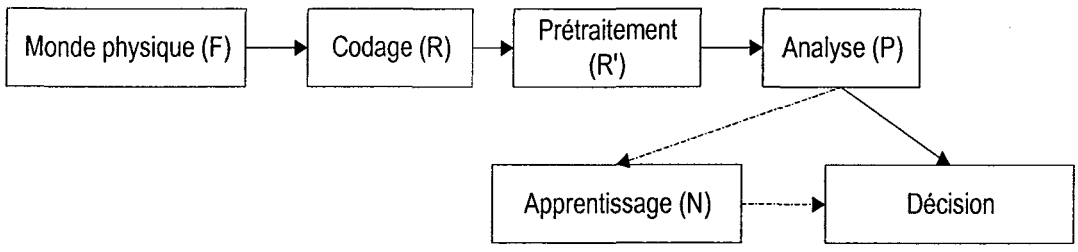


Figure 2.1 Schéma général d'un système de reconnaissance de formes

Pour le sujet de recherche qui nous occupe, les étapes du monde physique et du codage sont prises en charge par le FLOWCAM. Nous parlerons tout de même de ces deux

étapes pour rendre consistante la démarche. Nous conclurons sur les champs d'applications auxquelles cette discipline est classiquement rattachée, soit la reconnaissance de la parole, la reconnaissance de l'écriture et la vision.

2.2.1 Le monde physique

C'est celui qui est présenté dans sa forme la plus primaire, c'est-à-dire dont nous devons déterminer les caractéristiques les plus apparentes avant l'étape du codage. Le monde physique qui nous entoure est considéré comme un espace analogique de dimension **n** appelé l'espace des formes **F**. Le nombre de dimensions **n** est très grand pour cet espace. Il est particulièrement difficile de distinguer parmi toutes ces dimensions lesquelles représentent les particularités générales de l'objet d'étude. La loi de passage au monde discret nécessite alors une sélection et une simplification des paramètres qui sont nécessaires pour la classification de l'objet.

Imaginons un instant que l'on veuille classifier différentes espèces de champignons. Il apparaît clairement que les caractéristiques visuelles de longueur du pied, grosseur du pied, forme du chapeau, aplatissement du chapeau et couleur représentent des caractéristiques distinctives de l'identification des champignons. Nous pourrions également prendre d'autres caractéristiques (et il en existe de nombreuses) qui permettraient une classification plus fine. De toutes ces caractéristiques, il appert qu'il est possible de se limiter à quelques unes seulement pour une classification présentant peu de risques d'être inexacte.

Une caractéristique idéale pour la classification est celle dont la mesure est très semblable pour des objets de la même catégorie et très différente pour des objets de différentes catégories **[33]**.

2.2.2 Le codage

Le codage est une opération qui consiste en une conversion numérique du monde physique continu vers un monde physique discret. Cette discrétisation s'effectue à l'aide de capteurs branchés à une unité de conversion analogique-numérique. Ce sont des cartes d'acquisition de données qui font ce travail de conversion. Il est plus simple de travailler avec un signal numérique qu'avec un signal continu; les filtres numériques étant plus conviviaux et modifiables à volonté par programmation. De même, les systèmes de contrôle programmables offrent plus de possibilités de traitement à un signal numérique qu'à un signal analogique. On appelle espace de représentation (**R**) l'espace qui contient les paramètres servant à définir un objet. Cet espace contient généralement un nombre de dimensions encore trop important même s'il est fini. Il existe diverses méthodes permettant de distinguer parmi tous les paramètres, lesquels sont indépendants les uns des autres.

Dans MATLAB, il existe une fonction (prepca) permettant de conserver les composantes qui n'ont aucune covariance par rapport à d'autres paramètres tout en éliminant ceux présentant une forte covariance; cela s'apparente à une compression des données caractérisant les objets analysés. La dimension initiale **r** de cet espace est choisie volontairement grande de manière à pouvoir disposer d'un maximum d'information sur l'objet et ainsi pouvoir y sélectionner des sous-ensembles pour de multiples usages.

2.2.3 Le prétraitement des données

Comme les données sont obtenues à partir d'un appareil commercialisé, nous allons nous préoccuper essentiellement des étapes qui suivent. Le prétraitement des données est une étape importante car elle permet de sélectionner, dans l'espace de représentation, l'information nécessaire à l'application. Cette sélection passe par l'élimination du bruit dû aux conditions d'acquisition, par la normalisation des données, ainsi que par la suppression de la redondance. Le nouvel espace de représentation a une dimension plus petite mais demeure tout de même un espace de grande dimension. Les données subissant un prétraitement dans ce travail de recherche sont les images (provenant d'un microscope) et les paramètres cytométriques générés par le FLOWCAM. Il existe des outils dans MATLAB permettant de traiter toutes ces données de manière efficace.

La figure 2.2 schématise un système simplifié retrouvé généralement dans les applications de reconnaissance de formes.

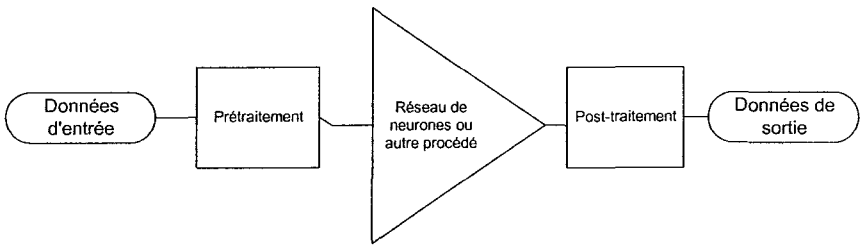


Figure 2.2 Schéma bloc des processus de traitement nécessaires en reconnaissance de formes

Nous présenterons à partir de la section 2.2.3.1 une vue d'ensemble des techniques de prétraitement qui existent à ce jour et qui ont une influence marquante dans les

applications quotidiennes. Ces techniques sont également valables pour le traitement des données à la sortie d'un système.

2.2.3.1 Prétraitement d'une image

On distingue généralement trois grandes classes d'opérations de prétraitement sur les images **[4]** :

- les manipulations d'intensité ou opérations ponctuelles;
- les opérations localisées ou sur voisinage;
- les opérations globales.

Les opérations ponctuelles modifient les valeurs de niveaux de gris des pixels de manière ponctuelle. On applique une fonction de transfert sur l'ensemble des pixels composant l'image.

$$\forall p \in E \Rightarrow L(p) = F(f(p)),$$ **où f(p) est la valeur du pixel p.** **Équation 2-1**

Dans les opérations sur voisinage, la nouvelle valeur du pixel est obtenue à partir des valeurs des points voisins. Chaque point de l'image est remplacé par une fonction ou combinaison linéaire des points voisins. Le voisinage est défini par une matrice dont les coefficients représentent un masque que l'on applique sur chacun des pixels. On appelle ce type de traitement local une convolution discrète. La librairie de traitement d'images de MATLAB contient des fonctions de conception de filtres à convolution discrète (passe-haut, passe-bas, filtres à gradients) de même que des algorithmes de segmentation et la possibilité de traitement spectraux sur les images.

Les traitements spectraux **[4]** entrent dans la catégorie des opérations globales car la transformation se fait sur la totalité de l'image et ce, pour chacun des pixels. En considérant l'image comme une fonction continue à deux variables f(x,y), sa transformée de Fourier, que l'on dénote par F(u,v) est donnée par l'équation suivante :

$$F(u,v) = \int_{-\infty-\infty}^{+\infty+\infty} f(x,y) e^{(2\pi i(ux+vy))} dxdy \qquad \textbf{Équation 2-2}$$

où u et v sont des variables fréquentielles. La transformée inverse de F s'écrit :

$$f(x,y) = F^{-1}(u,v) = \int_{-\infty-\infty}^{+\infty+\infty} F(u,v) e^{(-2\pi i(ux+vy))} dudv \qquad \textbf{Équation 2-3}$$

À partir de ces équations, on définit les quantités suivantes :

- le spectre de Fourier ou amplitude donnée par le module de la transformée F(u,v) : $|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{\frac{1}{2}}$ ou R est la partie réelle et I la partie imaginaire.
- Le spectre d'énergie : $E(u,v) = R^2(u,v) + I^2(u,v)$
- La phase : $\phi(u,v) = \tan^{-1}\left(\frac{I(u,v)}{R(u,v)}\right)$

Sous cette forme, la transformée de Fourier serait trop lente en vitesse de calcul à appliquer sur les images (pour un poste PC type de travail); on peut utiliser des algorithmes plus rapides comme la transformée de Fourier rapide ou FFT (*Fast Fourier Transform*) pour produire une transformation numérique.

Les fonctions de prétraitement comportent diverses transformations permettant d'obtenir une image ou un signal dénué d'informations superflues, la segmentation d'une image étant un des moyens de se débarrasser d'informations superflues ne servant nullement à caractériser un objet; on ne garde généralement que le contour de l'objet à considérer. Une fois l'image segmentée, on effectue un autre traitement consistant à extraire les caractéristiques indispensables à la reconnaissance. Les caractéristiques se doivent d'être idéalement invariantes en rotation, en translation et en homothétie **[3]**. C'est un passage obligé dans cette démarche classique lors de la détection d'un objet spécifique dans une image.

Les données caractéristiques sont un sous-ensemble de l'espace des formes. Un moyen pour obtenir les invariances d'homothétie, de translation et de rotation serait d'utiliser des configurations de réseau de neurones spécifiques **([35] page 320)** pour chacun des invariants mentionnés précédemment. Ce serait par contre un peu plus lourd en traitement car la base de données d'apprentissage doit être très élevée pour prendre en considération toutes les combinaisons possibles de translation, de rotation et d'homothétie. Il existe d'autres méthodes permettant de déterminer des caractères invariants, par exemple le calcul des moments d'ordre n **([35] page 322-323)** où $M_{20} + M_{02}$ est une quantité qui contient la propriété d'invariance en translation, rotation et homothétie. Cette dernière est précise mais demande tout de même un temps non négligeable en calcul. Ces deux moments représentent la longueur de l'axe majeur et de l'axe mineur de la forme en cause.

On retrouve une autre technique qui est en utilisation aujourd'hui et faisant référence à l'analyse spectrale. En effectuant une FFT (transformée de Fourier) sur l'image, on obtient rapidement des caractéristiques d'invariance de l'objet. C'est une méthode qui pourrait être utilisée dans l'approche expérimentale au même titre que les moments d'ordre **n** pour comparaison.

L'objet à être segmenté diffère généralement pour l'œil humain d'avec l'image de fond. Le contour de l'objet est une zone où le gradient est élevé par rapport aux zones du fond de l'image et du centre de l'objet. Ces changements de contraste sont détectés par des opérateurs matriciels qui calculent le gradient de l'image. Un opérateur classique utilisé couramment pour calculer le gradient d'une image est l'opérateur de Sobel (filtre horizontal et vertical) qui crée un masque binaire utilisant une valeur seuil définie par l'utilisateur. L'hypothèse derrière le filtre de Sobel est que si la mesure de contraste ou du gradient dépasse le seuil, on assigne à la coordonnées (x,y) une réponse positive indiquant à cet endroit la présence d'une discontinuité dans l'image et donc un point de contour de l'objet. Sur la figure 2.3, on peut voir les deux matrices représentant l'opérateur de Sobel vertical et horizontal.

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \text{ et } \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Figure 2.3 Exemple de masque (Sobel horizontal et vertical)

Cette matrice est appliquée sur chacun des pixels de l'image. Une fois les pixels traités, une nouvelle image se forme qui pourra être traitée de nouveau par un autre filtre dont la fonction fournira une autre image et ainsi de suite. Comme les filtres sont des opérateurs matriciels, il est recommandé de combiner tous les opérateurs de traitement en une seule matrice pour réduire le temps de calcul. L'utilisation des matrices est très répandue dans le traitement des images et des signaux. Le logiciel MATLAB permet de les manipuler aisément et rapidement.

Pour l'expérimentation, nous utilisons des filtres matriciels pour nettoyer et segmenter l'image. Pour ce faire, nous disposons dans MATLAB des fonctions « histeq » et « imfilter ». La fonction « imread » permet de lire l'image permettant aux fonctions « histeq » et « imfilter » d'opérer sur l'image. Nous allons voir par un exemple comment nous effectuerons ces étapes de filtrage et de segmentation d'objets dans une image.

2.2.3.2 Segmentation de l'image lors de la phase expérimentale

La segmentation joue un rôle très important dans la reconnaissance de formes. C'est à la suite de cette étape que nous calculons les moments d'ordres n pour ensuite procéder à l'extraction des paramètres. Nous avons construit cet exemple dans le seul but d'illustrer les processus permettant de segmenter convenablement une image. Nous utiliserons cet algorithme lors de l'étape expérimentale. Pour cet exemple, nous avons utilisé l'image d'une cellule végétale de phytoplancton (*Thalassionema_nitzschoides*) et en avons effectué la segmentation.

Il peut arriver que l'on ait plus d'une cellule dans l'image. Nous avons alors à séparer la cellule entière des autres cellules qui peuvent apparaître sur les bords de l'image, c'est-à-dire des cellules incomplètes. On énumère ici les étapes nécessaires pour segmenter une telle image.

- Étape 1 : Lecture de l'image;
- Etape 2 : Détection des contours internes et externes d'une cellule;
- Étape 3 : Linéarisation des contours des objets;
- Étape 4 : Dilatation des contours des objets;
- Étape 5 : Remplissage interne des objets;
- Étape 6 : Élimination des objets dépassant la bordure de l'image;
- Étape 7 : Adoucissement de l'objet et segmentation (contour externe seulement);

Voyons maintenant en détail le processus de chacune de ces étapes.

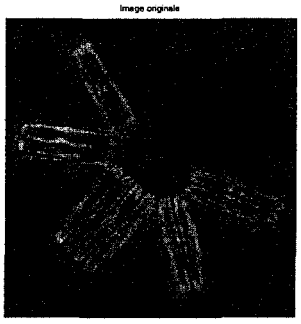
2.2.3.2.1 La lecture de l'image

La lecture d'une image avec MATLAB s'effectue à l'aide de la commande « imread ». Cette commande prend comme paramètre le nom du fichier contenant l'image.

```
Img=imread('s1.jpg');
Figure, imshow(img), title('Image originale') ;
```

La commande « Figure » permet d'ouvrir une fenêtre contenant l'emplacement vide réservé à un objet. La commande « imshow » permet d'afficher le contenu de la matrice

img représentant l'image sous forme numérique dans la fenêtre d'objet. La commande « title » permet de donner un titre à la fenêtre objet.



source de l'image : http://www.sb-roscoff.fr/Phyto/Phyto_gallery/Phyto_gallery.htm

Figure 2.4 Image originale d'un phytoplancton

2.2.3.2.2 Détection d'une cellule entière

À la figure 2.4, on retrouve l'image d'un organisme végétal pluricellulaire. Nous sommes en mesure de traiter cette image pour éliminer le plus possible les informations qui ne seront pas nécessaires lors de la phase d'apprentissage.

Avant de débuter un quelconque traitement sur cette image, nous devons ramener l'image couleur en format RGB en une image à teinte de gris. On utilise pour cela la commande « rgb2gray ». Cette image contient moins d'information concernant les couleurs mais a la particularité de conserver la forme de l'objet en niveaux de gris.

Imgray=rgb2gray(img) ;

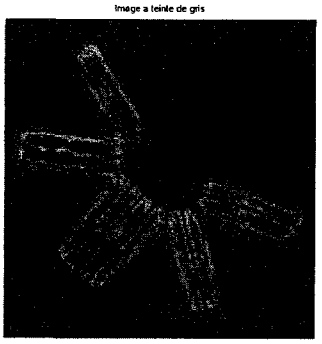


Figure 2.5 Image couleur transformée en image à niveau de gris

Maintenant, nous pouvons utiliser la commande « edge » avec comme filtre de gradient le filtre de Sobel, indiqué comme suit :

```
Bw1=edge(img2,'sobel',(graythresh(img2)*.1)) ;
Figure,imshow(Bw1),title('Masque à gradient binaire');
```

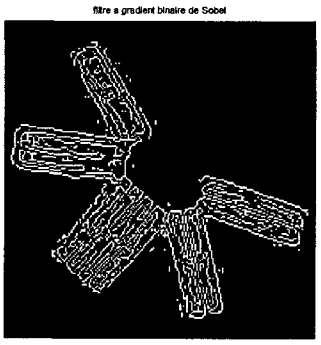


Figure 2.6 Image traitée par un filtre de Sobel pour détection du contour

Bien que le contour externe utile soit détecté, nous remarquons aussi la détection de plusieurs contours internes et indésirables. Ce qui importe dans notre expérience est la forme générale de l'objet, les caractéristiques internes étant obtenues à partir de paramètres cytométriques.

2.2.3.2.3 Linéariser les contours des objets

Le masque de gradient binaire montre les lignes à fort contraste dans l'image. Elles contiennent des discontinuités issues de l'image originale. Pour les adoucir, nous pouvons dilater horizontalement et verticalement l'image produite par le filtre de Sobel avec des éléments à structure linéaire. Ces éléments sont créés avec la fonction « strel ». On conserve les structures qui ont un minimum de trois pixels de linéarité.

```
ESL90=strel('line',3,90);
ESL0=strel('line',3,0);
```

2.2.3.2.4 Dilater les contours des objets

De façon à pouvoir fermer les contours des objets afin d'éviter des discontinuités de contours, nous disposons de la fonction « imdilate » en relation avec les matrices ESL90 et ESL0 construites à l'aide de la commande « strel ». De cette manière, l'objet sera bien identifié par un contour externe continu.

```
bwdil=imdilate(bw1,[ESL90 ESL0]);
figure,imshow(bwdil),title('Image dilate par élément structurel linéaire');
```

Un rehaussement de contraste est également effectué lorsque nous appliquons cette fonction à l'image. Il en découle une catégorisation plus nette des contours de l'objet.

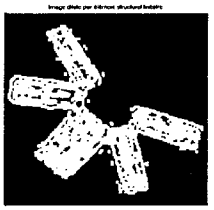


Figure 2.7 Image traitée par dilatation linéaire

À la figure 2.7, nous avons l'image résultante après dilatation. Nous remarquons que la forme est beaucoup plus distincte que dans l'image précédente. Le principe de la dilatation est de transformer une série de pixels colinéaires en une ligne continue fermée à épaisseur constante. Cette opération permet par la suite de remplir uniformément l'intérieur de l'objet sans dépassement du contour.

2.2.3.2.5 Remplissage interne du contour des objets

Si nous voulons prendre seulement le contour d'un objet pour fin de reconnaissance, nous devons éliminer les vides se trouvant à l'intérieur de celui-ci pour éviter d'avoir des éléments résiduels à l'intérieur. Ces éléments résiduels pourraient rendre la tâche d'isolation du contour plus ardu. Pour remplir l'intérieur d'un objet dont le contour a été tracé, on utilise la commande « imfill ». Cette commande prend comme paramètre la matrice de l'image dilatée suivi d'un second paramètre indiquant que l'on remplit les espaces vides (valeurs à 0) par des espaces pleins (valeurs à 1).

```
Bwdfill=imfill(Bwdil,'holes');
Figure, imshow(Bwdfill), title('Image binaire avec les vides remplis');
```

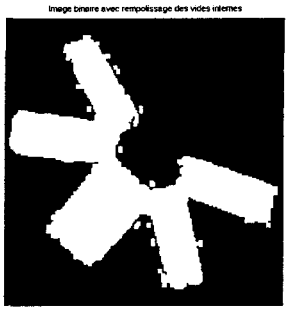


Figure 2.8 Remplissage des vides à l'intérieur du contour

La figure 2.8 nous montre que les vides sont remplis pour former une image pleine et que nous sommes maintenant prêts pour l'étape de segmentation qui consiste à donner une forme plus précise, contenant moins de pixels pour l'étape de prétraitement.

2.2.3.2.6 Élimination des objets qui touchent la bordure de l'image

Dans le cas où nous aurions d'autres objets se présentant autour de l'objet que l'on veut segmenter, nous pouvons les enlever à partir d'une commande qui permet d'éliminer tous les objets indésirables connectés sur les bordures de l'image. Cette commande se nomme « imclearborder ». Par son action, l'image est débarassée de tout objet qui ne doit pas être inclu dans la segmentation. On utilise le codage suivant :

```
bwnobord=imclearborder(bwdfill,4);
%figure,imshow(bwnobord),title('Image dilate par élément structurel linéaire');
```

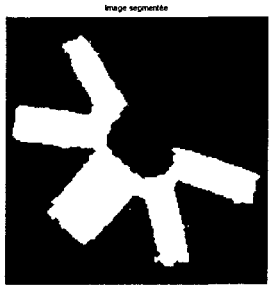


Figure 2.9 Remplissage des objets linéarisés.

2.2.3.2.7 Adoucissement et segmentation de l'objet

La segmentation de l'objet se termine par une commande d'adoucissement fournie par la commande « imerode » Cette commande est particulièment utile pour nettoyer les éléments indésirables se présentant autour de l'objet segmenté. Pour afficher uniquement le périmètre de l'objet excluant l'intérieur, on utilise la commande « bwperim ». La figure 2.9

ne contient plus qu'un seul objet qui sera traité lors de la segmentation. Une fois la segmentation terminée, nous disposons d'une image qui contient un objet à partir duquel on peut extraire des paramètres dimensionnels permettant de calculer les invariants associés à l'objet.

On présente en annexe le programme MATLAB complet qui effectue tous les traitements permettant la segmentation d'une image. Cette image serait ensuite présentée à un module de plus haut niveau comme un réseau de Kohonen pour l'étape de compression. Le traitement de l'image a permis de fournir une image présentant un nombre d'entrées avec le format binaire.

Le résultat final est montré dans l'image à la figure 2.10. Avec cette représentation minimale sans perte importante d'information, on tentera d'effectuer une reconnaissance de la forme par un réseau de neurones de manière efficace.

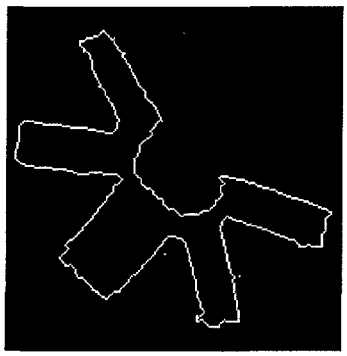


Figure 2.10 Image segmentée (périmètre de contour)

On peut également traiter cet objet pour en extraire les paramètres nécessaires à la classification en utilisant la technique des moments d'ordre n. Le calcul de ces moments

servira à déterminer des caractéristiques d'invariances qui seront présentées à un réseau de neurones; c'est la méthode qui sera utilisée lors de l'étape expérimentale.

2.2.3.3 Prétraitement d'un signal

La première étape de prétraitement d'un signal est d'effectuer une normalisation des données suivie d'un centrage. Nous verrons à la section 2.2.4 comment extraire les caractéristiques après le traitement de ce signal. Ce type de traitement facilite l'apprentissage du réseau de neurones (voir chapitre 4) car les données d'entrée ainsi que celles de sortie ont toutes le même ordre de grandeur. La normalisation dans MATLAB s'effectue à l'aide d'une fonction qui se nomme « premnmx ».

Cette fonction normalise les données entre –1 et +1 à l'entrée comme à la sortie. L'écart type est égal à un et la moyenne de l'ensemble des données est centrée à zéro. Regardons maintenant, du point de vue mathématique, comment s'effectue cette normalisation sur les données d'entrée ou de sortie.

On notera les symboles suivants ainsi que leurs définitions :

N nombre d'exemples valides de la table de données ;

q le nombre d'entrées ;

$\boldsymbol{X}^k$ le vecteur des entrées pour le $k^{ème}$ exemple ;

x_i^k la valeur de la $i^{ème}$ entrée pour le $k^{ème}$ exemple ;

$\boldsymbol{M}$ la matrice des entrées de dimensions (N,q) et de coefficient x_i^k ;

y^k scalaire représentant la valeur de la sortie désirée pour le $k^{ème}$ exemple ;

$\boldsymbol{Y}$ le vecteur de sortie de dimension $\boldsymbol{N}$ et de coefficient $\boldsymbol{y^k}$;

Les données normalisées sont calculées à partir des données brutes selon les relations suivantes :

$$x'^k_i = \left[\Omega_i x^k_i + \Sigma_i\right] \quad \text{et} \quad y'^k = \left[\Omega_s y^k + \Sigma_s\right] \qquad \textbf{Équation 2-4}$$

où

Ω_i : facteur multiplicatif pour l'entrée i;

Σ_i : facteur additif pour l'entrée i;

Ω_s : facteur multiplicatif pour la sortie;

Σ_s : facteur additif pour la sortie.

Pour déterminer les valeurs normalisées à partir des équations 2.4, nous devons d'abord calculer les valeurs moyennes et les écarts-types pour les entrées et les sorties.

Les moyennes sont données par :

$$M_i = \frac{\sum_k x^k_i}{N} \quad \text{et} \quad M_s = \frac{\sum_k y^k}{N} \qquad \textbf{Équation 2-5}$$

et les écarts-types sont donnés par les relations :

$$\sigma_i = \sqrt{\frac{\sum_k \left(x^k_i\right)^2}{N} - \frac{\left(\sum_k x^k_i\right)^2}{N^2}} \quad \text{et} \quad \sigma_s = \sqrt{\frac{\sum_k \left(y^k\right)^2}{N} - \frac{\left(\sum_k y^k\right)^2}{N^2}} \qquad \textbf{Équation 2-6}$$

Les facteurs multiplicatifs et additifs sont par le fait même définis comme :

$$\Omega_i = \frac{1}{\sigma_i} \quad \text{et} \quad \Omega_s = \frac{1}{\sigma_s} \qquad \textbf{Équation 2-7}$$

et

$$\Sigma_i = -\frac{M_i}{\sigma_i} \quad \text{et} \quad \Sigma_s = -\frac{M_s}{\sigma_s} \qquad \textbf{Équation 2-8}$$

Une fois les entrées et les sorties normalisées, il reste l'étape de l'extraction des paramètres ainsi que le choix de ceux-ci pour assurer la qualité de la classification. Cette étape permet le choix de paramètres invariants en rotation, en translation et en homothétie.

2.2.4 L'analyse

Le but de l'analyse est d'extraire les propriétés qui caractérisent l'objet et de les exprimer sous une forme numérique ou symbolique. Nous verrons dans le chapitre trois (3) une technique d'extraction de paramètres, soit la méthode PCA, qui donne, après traitement, un nombre restreint de caractéristiques ou paramètres tout en éliminant ceux de moindre importance. Que ce soit tant au niveau d'une image qu'au niveau d'un signal, les paramètres servent comme seules données représentant l'information sur un objet. Ceux-ci doivent être limités en nombre mais suffisamment nombreux pour être représentatifs de l'objet. La dimension de cet espace des paramètres est donc plus petite comparativement à l'espace du prétraitement. Il faut donc choisir les techniques les plus efficaces permettant d'extraire convenablement les bons paramètres car de ceux-ci dépendront de la bonne classification des objets présentés au système de classement.

2.2.4.1 Extraction des caractéristiques de forme pour l'image

Pour l'étape expérimentale, lors de l'analyse d'une image après que celle-ci soit segmentée, différents calculs seront nécessaires pour extraire les paramètres qui seront invariants tant en translation, en rotation et en homothétie. La méthode de calcul des moments **[8,9,35]** prend en compte l'organisation interne de la forme de l'objet.

Un moment élémentaire, M_{pq}, de l'image Ψ de l'objet, est défini par :

$$M_{pq} = \sum_{i=1}^{N}\sum_{j=1}^{N}\Psi(i,j)i^p j^q \qquad \textbf{Équation 2-9}$$

Habituellement, on se sert des dix premiers moments sur les seize (16) qui correspondent aux valeurs de $p \leq 3$ et $q \leq 3$. Le moment d'ordre 0, M_{00} est une valeur représentant la surface de l'objet.

Les moments d'ordre 1, M_{10} et M_{01}, définissent le centre de gravité (x_g, y_g) de la surface calculée précédemment :

$$x_g = \frac{M_{10}}{M_{00}} \quad y_g = \frac{M_{01}}{M_{00}} \qquad \textbf{Équation 2-10}$$

Les moments centrés d'ordre (p,q) sont définis par :

$$\overline{M}_{pq} = \sum_{i=1}^{N}\sum_{j=1}^{N} \Psi(i,j)(i - x_g)^p (j - y_g)^q \qquad \textbf{Équation 2-11}$$

On se servira des moments centrés d'ordre 2, $\overline{M}_{20}$, $\overline{M}_{02}$ **[35]** pour trouver les axes principaux d'inertie et, par le fait même, les allongements et orientations de la forme $\overline{M}_{11}$.

Pour le calcul des moments, nous avons:

$$\overline{M}_{20} = \sum_{i=1}^{N}\sum_{j=1}^{N} \Psi(i,j)(i - x_g)^2 \qquad \textbf{Équation 2-12}$$

$$\overline{M}_{02} = \sum_{j=1}^{N} \Psi(i,j)(j - y_g)^2 \qquad \textbf{Équation 2-13}$$

$$\overline{M}_{11} = \sum_{i=1}^{N}\sum_{j=1}^{N} \Psi(i,j)(i - x_g)(j - y_g) \qquad \textbf{Équation 2-14}$$

On pourra se servir de ces moments pour construire tous les paramètres nécessaires servant à caractériser un objet. Tous ces paramètres sont alors considérés comme des invariants. La combinaison $M_{20} + M_{02}$ selon **[35]** est supposée invariante sous translation, rotation et homothétie. Ce paramètre pourrait, à lui seul, caractériser une forme.

2.2.4.2 Filtrage et compression d'un signal

Nous incluons dans cette section une méthode permettant d'extraire des caractéristiques d'un signal ou d'une forme d'onde. Il existe des techniques développées et expérimentées dans **[37,38]**. Ces techniques sont efficaces mais demandent énormément de temps de calcul si nous avons affaire à un signal constitué de plusieurs milliers d'échantillons. Le signal fourni par le FLOWCAM n'est pas utilisable dans sa forme actuelle, seules des informations paramétriques sont disponibles. Il est possible que d'autres chercheurs puissent obtenir ce type de signal et ce qui suit permet de paramétrer le signal afin de servir de donnée d'entrée à un réseau de neurones.

Comme nous sommes en présence d'un signal fini non périodique, la technique d'analyse par ondelettes est alors appropriée pour filtrer et compresser un signal. Résumons en quelques pages ce qu'est la technique d'analyse par ondelettes. Si nous analysons un signal périodique bruité ou non, nous disposons de l'analyse par décomposition en séries de Fourier qui permet de connaître les fréquences caractérisant ce signal. Sur la figure 2.11, nous observons un signal quelconque et nous voulons connaître ses caractéristiques. La décomposition en série de Fourier donne l'information sur les fréquences qui compose ce signal comme on peut le voir à la figure 2.12.

Figure 2.11 Signal quelconque

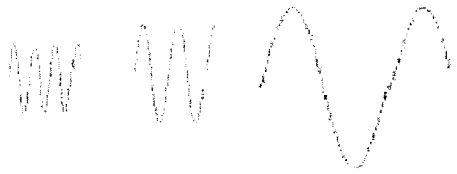


Figure 2.12 Décomposition en série de Fourier

Le problème rencontré dans ce type d'analyse est que les variations ou évènements qui se produisent dans le temps ne seront pas représentées ou détectées convenablement. Ils seront perdus ou dilués durant la transformation temps-fréquence. Le graphe de la figure 2.13 est un exemple de représentation d'une transformation temps-fréquence où l'amplitude est conservée mais dont le temps est remplacé par la fréquence.

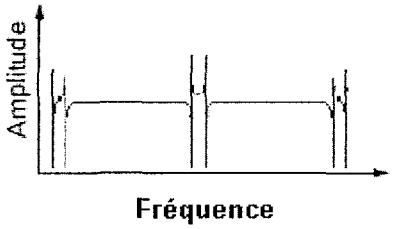


Figure 2.13 Représentation amplitude-fréquence

Dans l'analyse de Fourier, la transformation qui est appliquée à un signal temporel dont les dimensions sont l'amplitude et le temps produit un diagramme dont les dimensions sont l'amplitude et la fréquence. On voudrait obtenir un diagramme dont les paramètres sont le temps et la fréquence et pouvoir distinguer quand même des effets transitoires. Pour cela on considère une fenêtre de largeur fixe que l'on promène sur tout le signal tout en effectuant une décomposition en séries de Fourier du contenu de cette fenêtre. Cela donne un diagramme fréquence-temps comparable à celui de la figure 2.14. Le problème résiduel

dans ce type d'analyse est que nous devons toujours conserver la même résolution au niveau de la largeur de la fenêtre. On résout ce type de problème en utilisant l'analyse par ondelettes. Celle-ci permet de mettre en évidence des transitoires ou évènements temporels en variant la largeur de cette fenêtre selon différentes échelles dans le temps et dans la position de la fenêtre.

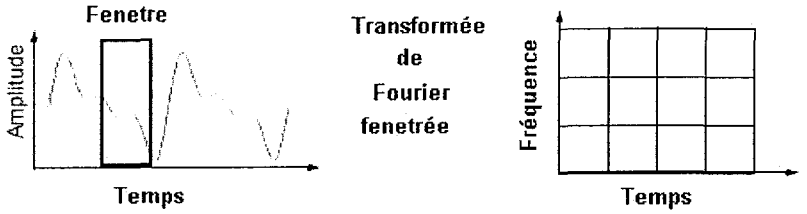


Figure 2.14 Transformée de Fourier à fenêtre temporelle

C'est une approche plus flexible mais dont le diagramme de transformation n'est plus constitué des dimensions temps-fréquence mais plutôt des dimensions échelle-temps.

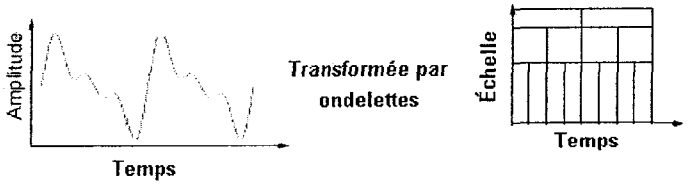


Figure 2.15Diagramme échelle-temps d'une transformée par ondelettes

Ce diagramme permet de reconnaître, selon les échelles, si le signal comporte des caractéristiques fréquentielles, temporelles ou les deux à la fois. On remarque également que selon le niveau d'échelle lors des transformations, on obtient une compression des données selon l'approximation voulue.

On utilise l'analyse par ondelettes pour comprimer des signaux ou des images. Dans le cas qui nous préoccupe, nous l'utilisons pour obtenir les coefficients de reconstruction du signal selon une certaine approximation. Plus nous augmentons le niveau d'échelle et moins grand sera le nombre de coefficients mais plus faible sera l'approximation.

Mathématiquement, le traitement par analyse de Fourier est représenté par la transformée de Fourier:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

Équation 2-15

D'une manière similaire, la transformée par ondelettes se sert d'une fonction finie comparativement à la transformée de Fourier qui utilise une fonction sinusoïdale infinie. On remarquera que cette la fonction Ψ est dépendante de l'échelle et de la position de l'onde dans le temps.

$$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt$$

Équation 2-16

On peut voir sur la figure 2.16 les formes d'ondes utilisées pour la transformée par ondelettes. Il existe d'autres formes d'ondelettes que nous retrouvons dans la littérature mais celles-ci sont moins efficaces que les ondelettes de Daubechies **[37]**.

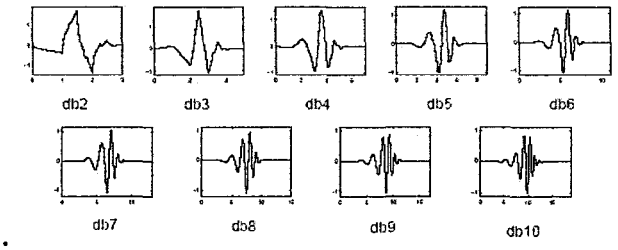


Figure 2.16 Ondelettes de Daubechies

Pour obtenir une décomposition convenable du signal, nous appliquons deux filtres dont l'un est un filtre passe-bas et l'autre un filtre passe-haut. On effectue ensuite un sous-échantillonnage à la sortie de chacun de ces filtres correspondant à la moitié du nombre d'échantillons du signal d'entrée.

Par exemple, le signal S de la figure 2.17 contient 1000 échantillons, les blocs A1 et D1 correspondent à la décomposition basse fréquence et haute fréquence respectivement. Le bloc A1 contiendra 500 échantillons de même que le bloc D1. En décomposant jusqu'à AAA3 et DAA3, nous obtenons respectivement 125 échantillons pour chacun de ces blocs.

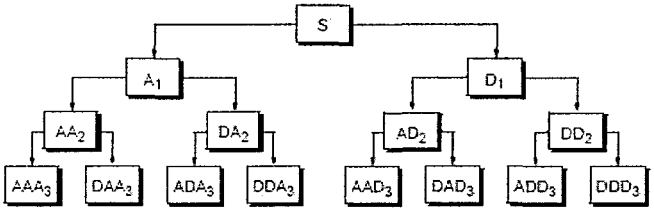


Figure 2.17 Arbre de décomposition du signal en coefficients d'ondelettes

À partir de la fonction inverse, nous aurons seulement besoin de 250 échantillons pour représenter le signal de 1000 échantillons, soit 125 échantillons de AAA3 et 125 échantillons de DAA3.

Nous pouvons descendre à un niveau tel que le signal traité contiendra beaucoup moins d'échantillons sans affecter les informations qui le caractérisent.

Lors d'une étape expérimentale, le signal sera traité à différents niveaux selon les caractéristiques de forme du signal. Une fois le niveau choisi, le signal résultant sera mémorisé à partir d'un réseau de neurones de type MLP ou par un réseau RBF qui jouera le rôle d'approximateur universel.

2.2.5 L'apprentissage.

L'apprentissage ou modélisation est une étape importante dans le processus de reconnaissance de formes. Son rôle est d'éclairer la décision à l'aide de connaissances à priori sur les formes. À partir de critères spécifiques aux formes, l'apprentissage tente de définir des classes de décision ou d'appartenance. La dimension **n** de l'espace des noms correspond globalement au nombre de modèle ou de classes existantes. Nous verrons dans le chapitre quatre (4) les techniques de classification qui définissent des classes d'appartenance par l'intermédiaire de modèles déterministes et statistiques. On utilise aussi des modèles stochastiques pour le classement des signaux.

2.2.6 La décision.

La décision ou classement est l'étape de reconnaissance proprement dite. Son rôle est de classifier la forme ciblée à partir de l'apprentissage réalisé. Les critères utilisés pour la décision sont habituellement les mêmes que ceux utilisés pour l'apprentissage.

Parmi les techniques utilisées, certaines sont fondées sur la notion de proximité et nécessitent de calculer une distance ou une probabilité de ressemblance avec les modèles définis. Dans les deux cas, la réponse peut être accompagnée d'un taux ou score de confiance.

2.3 Conclusion

Les applications découlant des techniques de reconnaissance de formes sont les suivantes : la reconnaissance de la parole, la reconnaissance de l'écriture, la vision artificielle ainsi que le contrôle de processus.

La reconnaissance de la parole est liée à plusieurs disciplines dont le décodage acoustico-phonétique, la reconnaissance de mots, la reconnaissance de phrases, la reconnaissance du locuteur et la compréhension du dialogue oral humain-machine. On retrouve dans les applications les plus courantes la commande vocale, la dictée automatique, la traduction en temps réel de langues étrangères et la rééducation de malentendants. Les progrès dans ce secteur d'activité sont continus et constants.

La reconnaissance de l'écriture qui est à ce jour une voie de recherche pleine de promesses est attachée à la reconnaissance des caractères manuscrits et imprimés, la reconnaissance de mots et de phrases, la reconnaissance du scripteur et la reconnaissance de documents. La technique Bayésienne est la plus courante à ce jour pour la reconnaissance des caractères. Des articles récents font référence à des applications atteignant jusqu'à 90% de reconnaissance sauf pour des textes présentant différentes polices de caractères. La recherche est également ouverte dans ce domaine.

Pour ce qui est de la vision artificielle, les travaux sont dirigés vers le traitement, l'analyse et l'interprétation des images. On retrouve une très grande variété de secteurs qui utilisent la vision artificielle tels que la médecine, la sécurité, l'industrie, la surveillance de processus en robotique, la géophysique avec l'analyse des images satellites, l'analyse du sol en pétrographie, la surveillance militaire, le guidage de cibles ainsi que d'autres que l'on oublie de citer.

Au niveau de la recherche sur le contrôle de processus, on utilise tout récemment des réseaux de neurones qui agissent en tant que filtre et outils de contrôle dont la gestion relève de la commande moderne. On utilise pour ce faire des filtres de Kalman qui agissent sur des variables non-observables pour maximiser des processus de contrôle. Des recherches récentes ont démontré leur utilité ainsi que leur fiabilité dans le contrôle de processus utilisés en robotique.

CHAPITRE 3

EXTRACTION DES CARACTÉRISTIQUES PRINCIPALES

3.1 Introduction

L'extraction des caractéristiques est une des étapes qui permettent de réduire considérablement le nombre de dimensions des vecteurs se présentant à l'entrée d'un réseau, diminuant par le fait même le nombre de neurones nécessaires à l'entrée de ce même réseau. En effet, chaque vecteur est constitué d'un nombre **n** de composantes et il existe des méthodes, telles que la régression linéaire multiple, l'analyse factorielle des correspondances, l'analyse discriminante et le réseau de Kohonen, pour n'en nommer que quelques unes, permettant de diminuer le nombre **n** de composantes tout en gardant l'information caractérisant l'objet à analyser sans perte d'informations significatives; le lecteur pourra consulter l'annexe pour un exemple pratique de cette technique. Lors de l'expérimentation, nous nous servons de cette méthode pour déterminer de manière optimale les paramètres qui serviront à caractériser chacune des cellules. On considère chacun des vecteurs comme un objet qui est présenté à l'entrée d'un réseau. Il est important de considérer le fait que dans le cas d'une application temps réel, le temps de traitement est un facteur non négligeable dans l'étape de reconnaissance.

Il faut que les données présentent une forme de corrélation entre elles pour que cette méthode de compression puisse fonctionner sinon le nombre de dimensions de l'espace caractéristique devra être diminué autrement.

Dans ce chapitre, la méthode PCA (Analyse par Composantes Principales) est présentée comme étant une technique d'usage très courant pour nettoyer l'ensemble des caractéristiques en éliminant toutes corrélations dans l'espace de formes. Nous verrons également comment, à partir d'un réseau de neurones, nous pouvons effectuer le même type de traitement.

3.2 La méthode PCA (Principal Component Analysis)

Dans l'essentiel, la méthode PCA est une technique permettant de réduire le nombre de dimensions de l'espace des paramètres ou des caractéristiques présentées à l'entrée d'un système pour fin de classification ou de traitement particulier. Le nuage de points faisant partie d'un espace à plusieurs dimensions est projeté dans un espace de dimension inférieure. L'orientation de cette projection aide à notre compréhension sur toutes relations existant entre chacun de ces points. Il est important de considérer le fait que la projection n'altère jamais l'orientation des points dans l'espace.

3.2.1 Explication géométrique

Une présentation imagée de la méthode nous permettra de comprendre son importance dans l'analyse des données. Cela nous amènera à mieux comprendre la présentation mathématique de cette méthode d'analyse. On utilisera au préalable une approche simplifiée que nous généraliserons par la suite.

Prenons l'image suivante, montrée à la figure 3.1, représentant des poissons de dimensions différentes.

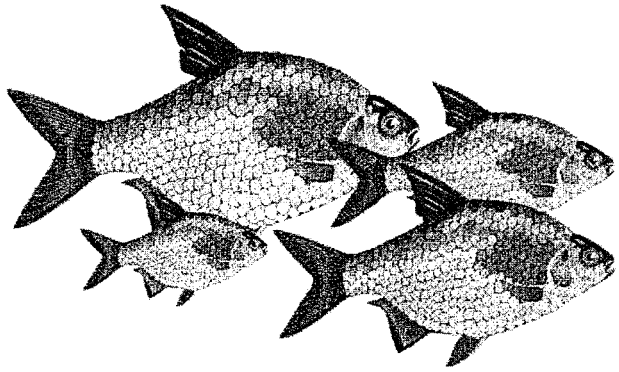


Figure 3.1 Collection de poissons à traits caractéristiques distincts

Pour différencier les poissons de cet ensemble ou collection, nous devons choisir des caractéristiques qui amèneront un pourcentage maximal de classification juste. À première vue, la longueur et la largeur du poisson semblent être des caractères distinctifs pour comparer la forme des poissons. L'image de la figure 3.2 montre les références de longueur et de largeur qui sont mesurées pour chacun des poissons.

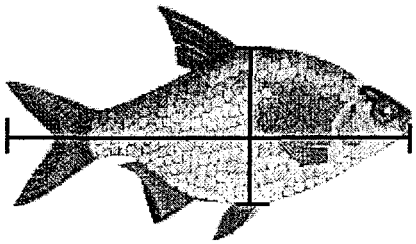


Figure 3.2 Représentation de la longueur et la largeur du poisson

Supposons maintenant que nous effectuons des mesures sur 50 poissons. Les mesures ainsi prises sont représentées par un graphe de points à la figure 3.3 ou l'axe horizontal représente la longueur tandis que l'axe vertical, la largeur.

À la vue de ce graphe, on remarque de manière évidente un rapport entre la longueur et la largeur, les poissons les plus longs tendant à être plus larges.

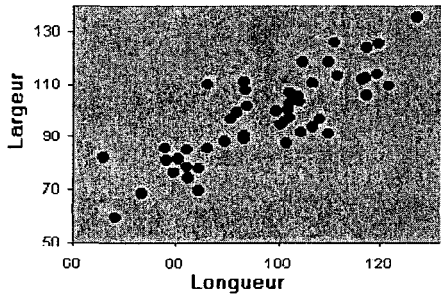


Figure 3.3 Graphe de points (x longueur-y largeur)

Déplaçons maintenant les axes de telle sorte que l'origine soit centrée sur le nuage de points, i.e. l'origine (0,0) devient maintenant centrée à la moyenne de x et de y. Cela représente un changement dans l'échelle de mesure.

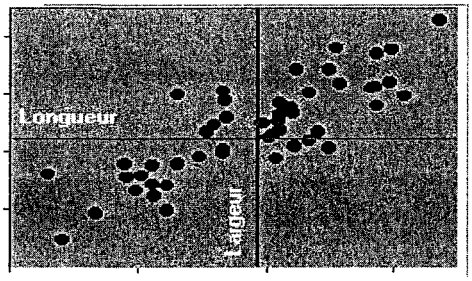


Figure 3.4 Déplacement des axes sur les moyennes

Il est primordial de se rendre compte que c'est le rapport entre les points que nous observons et non celui concernant les points avec les axes. Le déplacement des axes ne change pas la nature de la forme sous-jacente dans les données.

Dans l'image de la figure 3.5, on effectue une rotation des axes autour de l'origine. L'objectif est de positionner les axes de telle manière que l'on puisse obtenir la plus grande variance en longueur et en largeur.

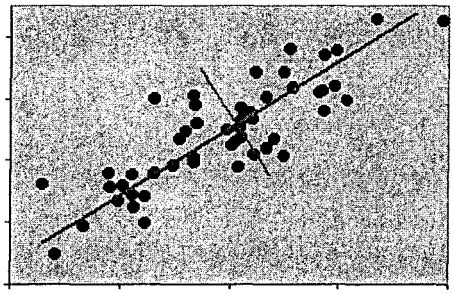


Figure 3.5 Rotation des axes selon la variance maximale

Les points forment alors un ellipsoïde dont les axes sont des combinaisons linéaires de la longueur et de la largeur dans des proportions différentes. Les plus grandes variations se retrouvent au niveau du grand axe incliné par rapport à l'horizontale. Cet axe est considéré comme la première composante principale qui est vu comme une projection des données, i.e. comme si on regardait les données sous un autre angle. L'axe majeur nous donne maintenant une nouvelle variable qui représente une forme de « dimension » du poisson. En effet, lorsque nous regardons à la gauche de cet axe, nous observons les petits poissons tandis qu'à la droite de l'axe majeur, nous observons les gros poissons, la forme étant conservée.

On voit également que cette nouvelle « dimension » du poisson est une combinaison linéaire des deux variables existantes avec des poids constants, soit

$$\text{Dimension} = w_1 * \text{longueur} + w_2 * \text{largeur} \qquad \textbf{Équation 3-1}$$

On peut faire en sorte que l'une des variables soit plus importante que l'autre en modifiant la valeur des poids. Supposons que nous considérions la longueur comme étant plus importante que la largeur dans la détermination de la dimension du poisson. Dans ce cas, nous pouvons utiliser des poids ou coefficients pour introduire une contribution plus importante pour une mesure que pour une autre. Par exemple

$$\text{Dimension} = 0.75 * \text{longueur} + 0.25 * \text{largeur} \qquad \textbf{Équation 3-2}$$

L'interprétation des poids est très importante dans toutes les méthodes d'analyse à plusieurs variables. Par exemple, assumons que la longueur et la largeur soient normalisées telle que chacune des variables a une moyenne égale à zéro et une variance égale à 1, nous pouvons alors conclure que la longueur est trois fois plus importante que la largeur dans la détermination de la dimension du poisson (coefficients de 0.75 pour la longueur et 0.25 pour la largeur). Si les deux variables ont des coefficients égaux (0.5), nous allons conclure que les deux variables sont d'égale importance dans la détermination de la grosseur du poisson. Si la longueur et la largeur n'étaient pas normalisées, il serait plus difficile de faire ressortir laquelle des variables contribue le plus sur la dimension du poisson.

La valeur des coefficients dépendra des échelles de mesure relative sur la longueur et la largeur. Supposons qu'en moyenne les valeurs sur la longueur soient quatre fois plus grandes que les valeurs de largeur. Pour les rendre d'égale importance, il faudrait que la valeur des poids soit de 0.2 pour la longueur et de 0.8 pour la largeur.

D'un point de vue pratique, nous pouvons disposer les points sur un nouvel axe des x correspondant à la variable « dimension » selon l'horizontale, cela donnant l'apparence que ce sont les points qui ont subit une rotation et non l'axe principal.

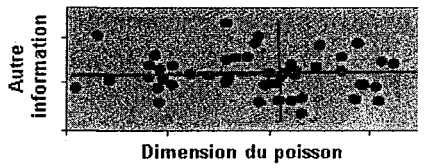


Figure 3.6 Représentation de la dimension des poissons

Que se passe-t-il pour l'axe vertical de l'ellipse? Celui-ci doit compter sur les variations restantes qui influencent peu sur la décision concernant la grosseur du poisson. Si cet axe était éliminé, nous perdrions un peu d'information sur la forme mais nous disposerions tout de même d'une grande quantité d'information qui représente un fort pourcentage sur la « dimension » des poissons. Avec cet exemple, on remarque que nous pouvons diminuer la dimension de l'espace des caractéristiques qui est de dimension 2 pour ne former qu'un espace de dimension un; de deux caractéristiques (longueur et largeur) nous n'en formons qu'une seule, soit la « dimension » du poisson. Nous verrons plus loin la façon de quantifier la perte d'information due à l'élimination d'une ou plusieurs dimensions de l'espace des caractéristiques.

Dans l'exemple précédent, les dimensions de longueur et de largeur sont fortement corrélées. Cependant, nous remarquons que l'une des valeurs propres de la matrice de corrélation est plus grande que l'autre. Par exemple, supposons que la corrélation longueur-largeur soit de 0.75, nous obtiendrons des valeurs propres de 0.25 et 1.75.

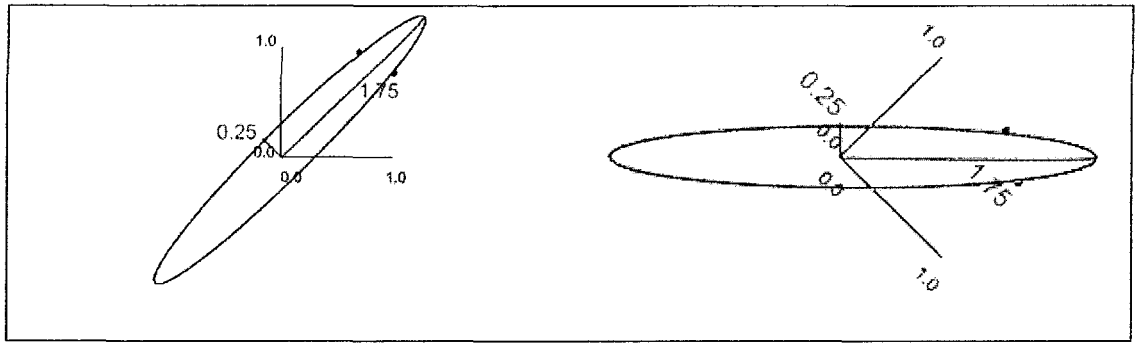


Figure 3.7 Orientations des vecteurs propres

Dans cet exemple, la longueur des axes ainsi que leur orientation est représentée par les vecteurs propres et les valeurs propres de la matrice de corrélation.

Si nous retenons seulement la variable « dimension », nous conservons 1.75/2.00 x 100 = {87.5%} de la variation originale. Par le fait même, en éliminant le second axe, nous perdons 12.5% de l'information originale.

Supposons que nous ayons maintenant une collection de poisson comme celle présentée à la figure 3.8 ; Pourrions nous représenter l'ensemble de ces poissons par la seule variable dimension? La réponse évidente est que l'on retrouve différentes dimensions et formes.

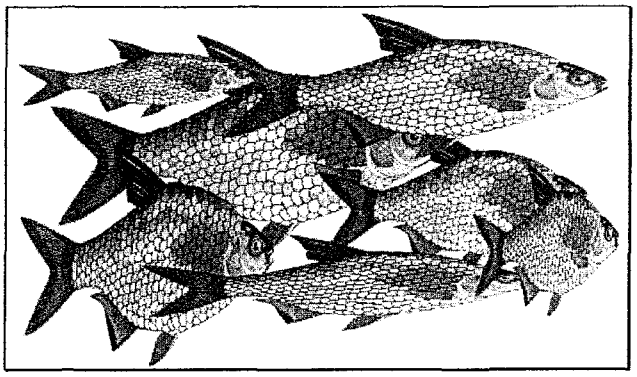
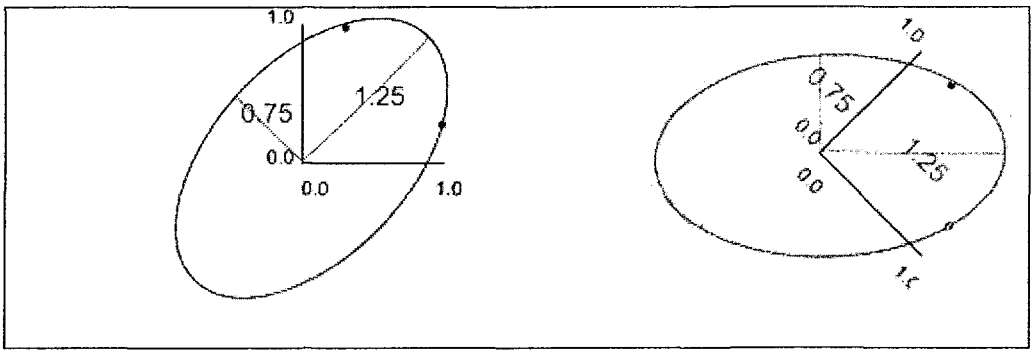


Figure 3.8 *Collection de poissons disparates que l'on tente de représenter par un seul paramètre*

En retenant seulement la variable dimension qui combine la largeur et la longueur, nous perdons l'information sur les différentes formes. Dans ces données, la longueur et la largeur ont un coefficient de corrélation faible de 0.25, donnant les valeurs propres suivantes :



Par conséquent, si nous retenons seulement le premier axe que nous nommons « dimension », nous conserverons seulement 1.25/2.00 x 100={ 62.5% } de l'information originale. Nous perdrions alors 37.5% de l'information représentée par la variable « forme ». Au lieu de prendre en considération un raisonnement a priori pour définir les contributions des variables originales sur les nouvelles variables, nous déterminons leurs

contributions à partir des vecteurs propres. Rappelons que les vecteurs propres décrivent l'orientation des nouveaux axes, relativement aux variables originales.

3.2.2 Développement mathématique de la méthode PCA

Nous allons ici détailler mathématiquement dans son ensemble cette méthode. Les procédures discutées dans cette section sont reliées sur les données d'entrée sans aucune référence aux données cibles. Cette méthode se compare à une forme d'apprentissage non supervisé. Notre objectif est de transformer des vecteurs $\boldsymbol{X^n}$ faisant partie d'un espace de dimension **d** $(x_1,....x_d)$ en vecteurs $\boldsymbol{Z^n}$ faisant partie d'un espace de dimension **M** $(z_1,....z_M)$ où **M<d**.

Nous noterons au passage que le vecteur $\boldsymbol{X}$ sera représenté, sans perte de généralité, comme une combinaison linéaire d'un ensemble **d** de vecteurs orthonormaux $\boldsymbol{u_i}$

$$X = \sum_{i=1}^{d} z_i u_i \qquad \textbf{Équation 3-3}$$

où les vecteurs $\boldsymbol{u_i}$ satisfont la relation d'orthonormalité.

$$u_i^T u_j = \delta_{ij}. \qquad \textbf{Équation 3-4}$$

Le symbole δ_{ij} représente le delta de Kronecker qui prend la valeur 1 si i=j et zéro autrement. Les vecteurs $\boldsymbol{u_i}$ sont dans la direction des vecteurs propres de la matrice de covariance. Ces vecteurs sont ordonnés selon l'ordre décroissant des valeurs propres correspondantes.

Le vecteur $\boldsymbol{u}_i$ est la composante principale et sa direction pointe vers la valeur propre la plus élevée. Comme la matrice de covariance est réelle et symétrique, les vecteurs propres sont orthogonaux. Ils forment une base naturelle de représentation des données.

Pour déterminer les coefficients z_i, on utilise l'équation 3.3, ce qui donne

$$z_i = u_i^T X \qquad \textbf{Équation 3-5}$$

L'équation 3.5 peut être vue comme une simple rotation du système de coordonnées représenté par les paramètres x en un système de coordonnées dont les données sont maintenant représentées par les paramètres z.

Supposons maintenant que nous retenions seulement un sous-ensemble M<d des vecteurs de base $\boldsymbol{u}_i$, tel que nous utilisons seulement M coefficients z_i. Les coefficients restants seront remplacés par des constantes $\boldsymbol{b}_i$ et chaque vecteur X sera approché par l'expression suivante :

$$\tilde{X} = \sum_{i=1}^{M} z_i u_i + \sum_{i=M+1}^{d} b_i u_i \qquad \textbf{Équation 3-6}$$

Dans cette relation, on voit très bien la réduction sur les composantes du vecteur X qui est passé de d dimensions à M dimensions ou degrés de liberté. On notera également que les valeurs des z_i sont différentes pour chaque échantillon alors que les $\boldsymbol{b}_i$ sont constants.

Considérons maintenant une collection d'objets représentés par N vecteurs X^n où n=1...N. Nous voulons choisir une base de vecteurs u_i et des constantes b_j telles que l'approximation donnée par l'équation 3.6 avec les valeurs de z_i déterminées par l'équation 3.5 puisse donner la meilleure approximation du vecteur X en moyenne pour tout l'ensemble des données.

L'erreur dans l'approximation de X^n introduite par la réduction de paramètres ou de dimensions est donnée par

$$X^n - \tilde{X}^n = \sum_{i=M+1}^{d} (z_i^n - b_i) u_i \qquad \textbf{Équation 3-7}$$

On peut définir la meilleure approximation sur l'ensemble des vecteurs en minimisant la somme du carré des erreurs sur toute la collection de données. Nous minimiserons alors la relation suivante

Équation 3-8

$$E_M = \frac{1}{2}\sum_{n=1}^{N} \left\| X^n - \tilde{X}^n \right\|^2 = \frac{1}{2}\sum_{n=1}^{N} \left(\sum_{i=M+1}^{d} (z_i^n - b_i) u_i \cdot \sum_{i=M+1}^{d} (z_i^n - b_i) u_i \right) = \frac{1}{2}\sum_{n=1}^{N} \sum_{i=M+1}^{d} (z_i^n - b_i)^2$$

où nous utilisons la relation d'orthonormalité de l'équation 3-4. En égalant à zéro la dérivée de E_M par rapport à b_i, nous trouvons que

$$b_i = \frac{1}{N}\sum_{n=1}^{N} z_j^n = u_i^T \overline{X} \qquad \textbf{Équation 3-9}$$

où $\overline{X}$ représente la moyenne des vecteurs X.

$$\overline{X} = \frac{1}{N}\sum_{n=1}^{N} X^n \qquad \textbf{Équation 3-10}$$

Utilisant l'équation 3.5 et l'équation 3.9, nous pouvons écrire la somme des erreurs au carré comme

$$E_M = \frac{1}{2}\sum_{i=M+1}^{d}\sum_{n=1}^{N}\left\{u_i^T\left(X^n - \overline{X}\right)\right\}^2 = \frac{1}{2}\sum_{i=M+1}^{d} u_i^T \Sigma u_i \qquad \textbf{Équation 3-11}$$

où Σ est la matrice de covariance.

Dans le logiciel MATLAB, on retrouve une fonction qui permet de traiter les données par la méthode PCA. Cette fonction porte le nom de **prepca**. Dans l'annexe A, on retrouve le code MATLAB qui effectue une analyse par composantes principales.

3.3 Le réseau de Kohonen

Le réseau de Kohonen est un réseau de neurones dont la particularité est d'agir en tant que compresseur de données, i.e. en conservant uniquement les informations caractérisant l'objet présenté au réseau sans perte importante d'information. Une élimination des paramètres corrélés s'effectue comme avec la méthode PCA. En effet, sa capacité de conservation topologique permet une réduction des données de l'entrée selon le nombre de neurone formant le réseau. Nous verrons plus en détail dans le chapitre quatre (4) le procédé détaillé de ce type de réseau.

3.4 Utilisation de la méthode PCA lors de l'étape expérimentale

Lors de l'étape expérimentale, nous utilisons la méthode PCA pour déterminer le plus grand nombre de paramètres orthogonaux pouvant caractériser la forme de la cellule sous étude. Les paramètres caractérisant l'image ainsi que les paramètres cytométriques caractérisant les signaux de fluorescence (Chlorophyle et phycoéritrine) seront ceux qui seront traités par la méthode PCA. Par exemple, si nous choisissons 10 paramètres que nous croyons être ceux caractérisant l'image d'une cellule en plus de quatre paramètres caractérisant les signaux cytométriques et que nous soumettons ceux-ci à un traitement par la méthode PCA, il y a de fortes chances que nous passions de 10 à 5 paramètres orthogonaux, laissant sous-entendre que les paramètres que nous avons choisis ne sont pas tous nécessaires pour caractériser la cellule.

Nous nous servirons donc de ce fait pour construire un tableau contenant cinq listes de paramètres dont nous savons qu'il y a de fortes possibilités que ceux-ci caractérisent l'objet d'analyse et, à partir de la méthode PCA, découvrir la liste qui représente une plus faible corrélation. Nous allons également faire varier le paramètre de variance en vérifiant si les paramètres orthogonaux déterminés par la méthode PCA sont trop proches ou non entre eux. Dans le chapitre six (6), nous élaborons sur cette méthode de recherche des meilleurs paramètres et nous discutons de sa fiabilité.

3.5 Conclusion

La méthode d'analyse par composantes principales (PCA) permet de connaître les rapports existant entre les différentes dimensions d'un ensemble de données. Comme cette technique permet de diminuer la quantité de données originales en abaissant le nombre de dimensions tout en limitant la perte d'information, elle semble très prometteuse pour les applications de classification effectuées par des réseaux de neurones. En effet, plus les données sont optimisées à l'entrée d'un réseau, plus rapidement s'effectuera la classification et plus rapide sera la décision du système. Une des limitations de cette technique est son approche linéaire; si les données sont entachées de bruit, la méthode PCA présente quelques problèmes au niveau de la recherche de la variance maximale. Les axes contenant les informations dominantes sont pris en considération tandis que ceux présentant une moindre influence sont éliminés. Lors de l'expérimentation, on utilise la méthode PCA pour diminuer l'espace des paramètres. Le réseau de Kohonen pourrait également être utilisé comme outil de décorrélation mais ne sera pas expérimenté dans cette recherche.

CHAPITRE 4

LES TECHNIQUES DE CLASSIFICATIONS

4.1 Introduction

Cette section présente une synthèse théorique et pratique de diverses techniques de classification et d'approximateurs universels (approximation d'une fonction dans un espace à n dimensions). On discutera plus particulièrement du réseau à couches multiples MLP (Multi Layer Perceptron) en association avec la méthode d'apprentissage de rétropropagation du gradient de l'erreur, communément appelée « back-propagation ». C'est, à ce jour, la technique qui donne les meilleurs résultats au niveau de la classification en utilisant l'apprentissage supervisé. Le réseau RBF (Radial Basis Function) à apprentissage supervisé est également discuté en détail pour ses qualités de génération de fonctions et d'approximateur universel.

Avant d'examiner les techniques utilisées, nous allons résumer le fonctionnement d'un neurone en commençant par sa nature biologique et terminant par la modélisation qui en est une pâle approximation et porte le nom de neurone formel. De par cette modélisation, diverses méthodes de classification ont été développées. Les fonctions discriminantes linéaires sont une de ces techniques qui se généralisent aux fonctions discriminantes non linéaires.

Cette théorie de base vient ensuite ouvrir la voie à la théorie des RNA (Réseau de Neurones Artificiels) à couches multiples à connexions directes telle que le réseau MLP. Nous verrons le perceptron multi-couches avec diverses méthodes de convergence; par exemple la méthode de Levenberg-Marquardt **[35]** qui est intéressante du point de vue de la vitesse de convergence. Le seul défaut de cette méthode de convergence est le calcul approximatif de sa matrice hessienne pour des réseaux non-linéaires.

Une problématique importante se situe au niveau de la configuration des réseaux de neurones dont la théorie est inexistante jusqu'à ce jour. C'est un domaine de recherche largement ouvert mais une nouvelle technique provenant des plans d'expériences permettrait toutefois d'optimiser la configuration d'un réseau de neurones selon le contexte du problème, une documentation de cette méthode se retouve dans la section bibliographique **[30,31]**. Nous inclurons une courte partie théorique sur les plans d'expériences ainsi que la méthode Taguchi qui servira à optimiser la configuration du réseau de neurones utilisé lors de l'étape expérimentale. On discutera également de quelques méthodes de classification par apprentissage non-supervisé sans toutefois les expérimenter dû à l'énorme quantité de données nécessaires pour une recherche efficace de classes indépendantes. Le désavantage évident de cette méthode d'apprentissage se présente lors de l'insertion d'une nouvelle donnée représentant une nouvelle classe. Le temps d'apprentissage augmente alors très rapidement à cause de la recherche de nouvelles classes et n'est donc pas applicable pour des systèmes fonctionnant en temps réel.

4.2 Le neurone biologique

Un neurone ou cellule nerveuse est une cellule biologiquement spécialisée dans le traitement de l'information.

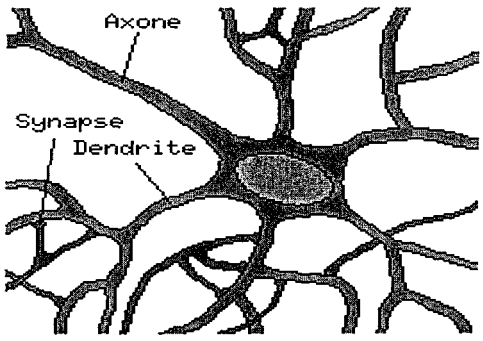


Figure 4.1 Neurone biologique

http://www.chimique.usherbrooke.ca/cours/gch445/neurones-intro.html

La figure 4.1 représente un tel neurone. On retrouve dans la cellule nerveuse un corps cellulaire ou soma et deux types de prolongements cellulaires plus ou moins ramifiés : les dendrites et l'axone. Le soma possède un noyau qui contient l'information génétique de l'organisme dont le neurone est issu, et un cytoplasme qui contient la « machinerie de synthèse » des protéines constitutives ou régénératrices de ce neurone (membrane et squelette - cytosquelette - de la cellule, mais aussi neurotransmetteurs, ...). Un neurone reçoit des signaux (impulsions) provenant d'autres neurones au niveau des dendrites (« pôle récepteur » de la cellule) et transmet l'information, générée dans son corps cellulaire, via l'axone (transmission et « pôle émetteur » de la cellule). L'extrémité de l'axone peut être plus ou moins ramifiée en une arborisation terminale. Chaque branche de cette arborisation est composée d'une terminaison autour de laquelle se trouvent les synapses. Une synapse est une structure élémentaire et une unité fonctionnelle de communication (axo-

dendritique) entre deux neurones. Lorsque l'impulsion nerveuse (potentiel d'action) atteint la synapse, certaines substances chimiques (acides aminés, protéines), appelés neurotransmetteurs sont libérées dans l'espace synaptique séparant la membrane axonale (du premier neurone) de la membrane dendritique (du second neurone). Le neurotransmetteur traverse l'espace synaptique et se fixe sur des sites récepteurs post-synaptiques du neurone récepteur. Selon que la synapse est excitatrice ou inhibitrice, cette action du neurotransmetteur va potentialiser, ou au contraire, inhiber la capacité du neurone récepteur à émettre des potentiels d'action. En fait, le fonctionnement synaptique va s'ajuster en fonction des signaux reçus et envoyés, ce qui permet de dire que les synapses apprennent leur comportement à partir des activités dans lesquelles elles sont impliquées (ou à partir des signaux qui leur sont imposés...). Ce comportement synaptique façonné par l'histoire de la synapse fonctionne donc de manière similaire à une mémoire et pourrait même être éventuellement responsable de la mémoire humaine.

Chez l'humain, le cortex cérébral est une pellicule de neurones d'environ 2 à 3 millimètres d'épaisseur et d'une surface de l'ordre de 2200 cm^2, c'est-à-dire environ deux fois la surface d'un clavier. Le cortex contient environ 10^{11} neurones, représentant approximativement le nombre d'étoiles se trouvant dans la voie lactée. Les neurones sont interconnectés entre eux de manière bien plus complexe et bien plus dense que ne l'est par exemple un réseau téléphonique. On évalue le nombre de connexions entre un neurone et ses voisins entre 10^3 et 10^4, ce qui fait que le cortex humain contient entre 10^{14} et 10^{15} connexions.

Les neurones communiquent en émettant des trains de potentiels rapides et très courts de l'ordre de quelques millisecondes. Le message est modulé en terme de fréquences qui peuvent varier de quelques hertz à quelques centaines de hertz, soit un million de fois plus lentement que la plus rapide bascule dans un circuit électronique actuel. Pourtant les décisions perceptives complexes, telles que la reconnaissance des visages, se font en quelques centaines de millisecondes sur un réseau dont la vitesse est de l'ordre de la milliseconde. Ceci indique que l'évaluation ne nécessite qu'une centaine d'étapes sérielles. En d'autres termes, notre cerveau, dans de telles tâches perceptives, exécute des programmes qui n'excèdent pas la centaine d'instructions parallèles. C'est ce que l'on appelle la règle des 100 pas **[27]**. Des considérations temporelles du même ordre montrent que la quantité d'information transmise d'un neurone à l'autre doit être très petite (quelques bits), et donc que l'information critique n'est pas transmise directement, mais qu'elle est contenue et répartie dans les inter-connexions, d'où le nom de modèle connexionniste et mémoire associative pour décrire les RNA.

4.3 Les réseaux de neurones artificiels (RNA)

Les réseaux de neurones artificiels ou RNA sont des assemblages fortement connectés d'unités de calcul. Chacune des unités de calcul est un neurone formel qui est, en soi, une formulation mathématique ou un modèle très simplifié d'un neurone biologique. Nous verrons que les RNA ont des capacités de mémorisation, de généralisation et d'une certaine forme d'apprentissage. On classe généralement les réseaux de neurones en deux catégories: les réseaux faiblement connectés à couches que l'on appelle des réseaux « feed-forward » ou réseaux directs et les réseaux fortement connectés que l'on appelle des

réseaux récurrents. Dans ces deux configurations, on retrouve des connexions totales ou partielles entre les couches. Le perceptron multicouches et le réseau RBF (Radial Basis Function) appartiennent à la catégorie des réseaux « feed-forward », l'information transigeant uniquement de l'entrée vers la sortie (le réseau RBF étant de plus un réseau à architecture évolutive) tandis que les réseaux de Hopfield et ceux de Elman appartiennent à la catégorie des réseaux récurrents. Le réseau de Kohonen est un autre exemple de réseau à connexions récurrentes. Il peut être utilisé tant pour la classification, la compression de données ou dans le contrôle de systèmes complexes en automatisme.

Pour chacun de ces types de réseaux, il existe également des méthodes ou des algorithmes d'apprentissage permettant une modification des paramètres de poids et de biais. Les paramètres de poids et de biais caractérisent la position des hyperplans qui séparent toutes les classes On retrouve plusieurs algorithmes d'apprentissage tels que l'algorithme du perceptron, la règle d'apprentissage de Hebb, la règle d'apprentissage de Widrow-Hoff (règle Delta) ainsi que des algorithmes généralisés de Widrow-Hoff pour les réseaux multicouches (rétropropagation du gradient de l'erreur...). On peut classer en deux catégories les formes d'apprentissage, soit l'apprentissage en mode supervisé et l'apprentissage en mode non supervisé; chacun ayant ses qualités et défauts.

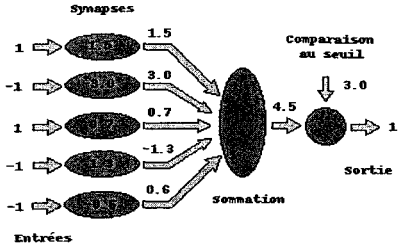
Pour être en mesure d'utiliser les RNA, nous devons préparer les données qui se présentent en entrée et en sortie. Les étapes importantes qu'il ne faut pas considérer à la légère sont les étapes de prétraitement et de post-traitement. Les étapes de prétraitement améliorent considérablement la précision de la classification tandis les étapes de post-

traitement permettent de donner une autre forme aux données de sortie. Nous donnerons quelques précisions à ce sujet tout au long de ce document.

4.3.1 Le neurone formel

Le développement des RNA a été possible grâce à la détermination et aux recherches avant-gardistes de McCulloch et Pitts, deux chercheurs en neuro-biologie qui étudièrent le fonctionnement du cerveau humain. C'est en 1943 que McCulloch et Pitts **[40]** proposèrent le premier modèle selon les observations qu'ils firent sur les neurones biologiques. Voici ce qu'ils avaient remarqué : « Un neurone formel fait une somme pondérée des potentiels d'actions qui lui parviennent (chacun de ces potentiels est une valeur numérique qui représente l'état du neurone qui l'a émis), puis s'active suivant la valeur de cette sommation pondérée. Si cette somme dépasse un certain seuil, le neurone est activé et transmet une réponse (sous forme de potentiel d'action) dont la valeur est celle de son activation. Si le neurone n'est pas activé, il ne transmet rien. »

Le premier modèle réalisé en laboratoire fut un automate binaire ou à seuil qui représentait le neurone formel de McCulloch et Pitts (voir figure 4.2).



Courtoisie de l'université de Sherbrooke

Figure 4.2 Neurone formel à plusieurs entrées avec seuil de comparaison β=3.0

La fonction de seuil utilisée dans ce modèle s'apparentait à la fonction suivante :

$$f(x)=1 \quad si\ x>\beta \quad et \quad f(x)=0 \quad si\ x\leq\beta \qquad \textbf{Équation 4-1}$$

Le neurone formel est un modèle mathématique simplifié du neurone biologique. Le neurone formel est constitué de plusieurs entrées dont chacune est multipliée par un poids. L'ensemble des résultats de chacune des entrées est présenté à un sommateur pour en former une combinaison linéaire. À la figure 4.3, on schématise un neurone formel avec ses entrées, ses poids ainsi que le module de sommation qui pondère les entrées.

On appelle apprentissage la recherche de la valeur qui doit être donnée à chacun des poids pour permettre au réseau de classifier convenablement des données. Les méthodes d'apprentissage les plus courantes est l'apprentissage supervisé et non-supervisé.

Les seules opérations permises pour ce type de neurone sont les opérations **ET**, **OU**, et **NON**, des opérations qui permettent de résoudre uniquement des problèmes linéairement séparables.

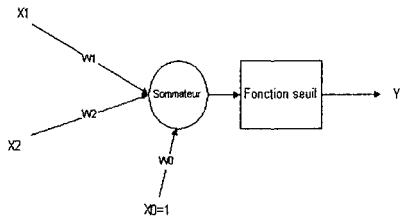
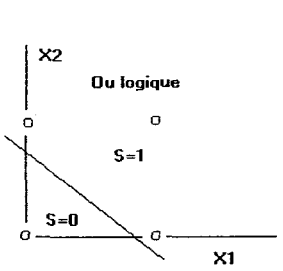


Figure 4.3 Neurone formel binaire de McCulloch-Pitts.

Pour bien comprendre le fonctionnement d'un neurone formel, nous allons présenter quelques exemples liés aux fonctions logiques. Nous utiliserons, pour ce faire, les fonctions **ET**, **OU** qui, comme nous le verrons, sont des fonctions linéairement séparables et la

fonction logique **XOR** (ou exclusif) qui se trouve être une fonction qui n'est pas linéairement séparable par un neurone formel. Nous verrons comment l'ajout d'une seconde couche peut résoudre ce type de problème qui se rencontre fréquemment dans les applications quotidiennes.



X1	X2	Sortie
0	0	0
0	1	1
1	0	1
1	1	1

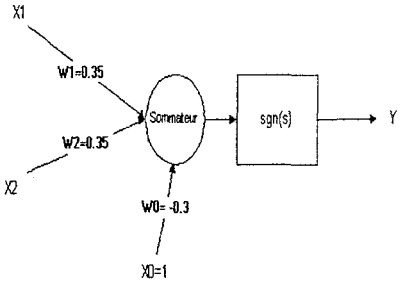
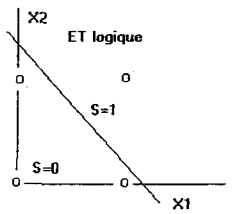


Figure 4.4 OU logique : table de vérité et valeur des poids du neurone formel.

Pour comprendre le **OU** logique, on présente celui-ci sous forme de table de vérité. On peut voir, sur la figure 4.4, les valeurs des entrées et de la sortie pour chacun des états. Les valeurs des poids se stabilisent après apprentissage. Si on présente sur X_1 et X_2 la valeur zéro, on veut à la sortie du sommateur le résultat suivant: $0.35(0)+0.35(0)-0.3=-0.3<0,\ Y=0$. Si la fonction **sgn(s)** donne 1 pour S>0 et 0 pour S<0, alors la sortie Y pour cet état donnera la valeur 0, ce qui correspond au comportement de la porte **OU**. Si nous appliquons maintenant X_1=1 et X_2=0 à la sortie du sommateur, nous obtenons $0.35(1)+0.35(0)-0.3=+0.05>0,\ Y=1$. Nous aurons le même résultat pour X_1=0 et X_2=1. Si X_1 et X_2 sont tous deux égaux à un (1), la sortie vaudra $0.35(1)+0.35(1)-0.3=+0.4>0,\ Y=1$.

Pour ce qui est de la porte **ET**, avec des valeurs différentes de poids, nous obtenons les sorties désirées pour chacun des états de la porte **ET**.



X1	X2	Sortie
0	0	0
0	1	0
1	0	0
1	1	1

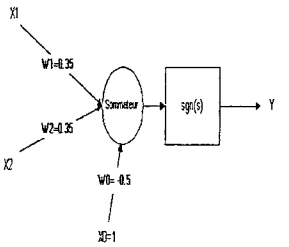
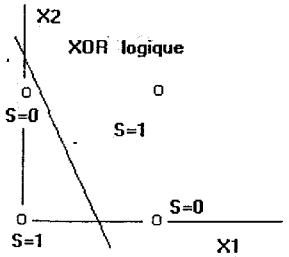


Figure 4.5 ET logique : table de vérité et valeur des poids du neurone formel

Pour les fonctions logiques **OU** et **ET**, il n'y a aucun problème au niveau de la séparabilité des classes. Par contre, un neurone formel ne pourra résoudre un problème de type **XOR**. Comme on peut le voir à la figure 4.6, la seule manière d'arriver à résoudre ce type de problème est d'ajouter une nouvelle couche contenant un neurone. C'est ce que Minsky et Pappers conclurent dans l'article qu'ils publièrent en 1970 **[41]**.



X1	X2	Sortie
0	0	0
0	1	1
1	0	1
1	1	0

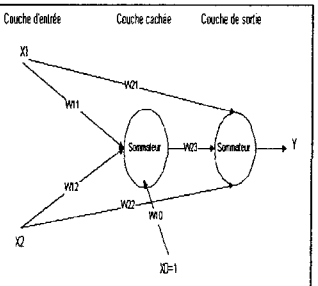


Figure 4.6 Problème de séparabilité pour le XOR logique

A la figure 4.7, on a une droite de séparation qui ne peut séparer les deux classes. Pour résoudre ce problème, Minsky et Papper proposèrent une configuration en multicouches mais il restait à trouver comment effectuer l'ajustement des paramètres de poids et de biais dans les couches cachées

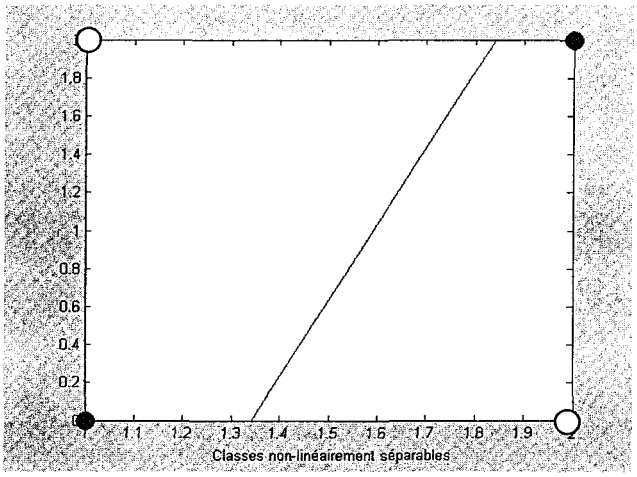


Figure 4.7 Classes non séparables pour un perceptron

Entre les années 1970 et 1980, plusieurs chercheurs travaillèrent à la résolution de ce problème et découvrirent finalement un algorithme qui permettait de résoudre ce problème de configuration des poids et des biais. Cela ouvrit la porte à une véritable révolution dans l'utilisation des RNA (Réseau de Neurones Artificielles) dans toutes les sphères d'activités.

Les réseaux à couches multiples que l'on nomme MLP (*Multi-Layer Perceptron*) seront étudiés un peu plus loin ainsi que la règle d'apprentissage de rétropropagation du gradient de l'erreur. On peut voir sur la figure 4.8 la couche d'entrées Xi, Yi représentant la couche cachée et finalement la couche de sortie Zi.

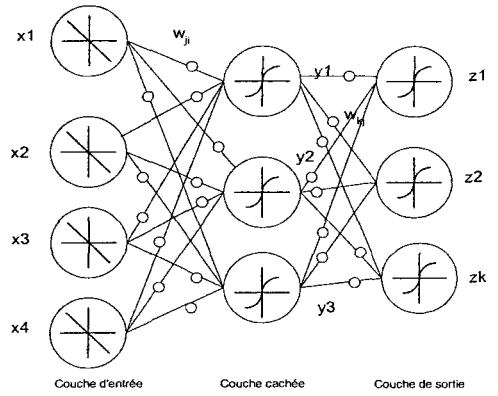


Figure 4.8 Réseau MLP à connections directes

Le neurone formel, de par sa nature, agit donc comme un séparateur à fonction discriminante linéaire. L'ajout de plusieurs neurones à la couche d'entrée produit un discriminateur linéaire complexe. Dans le cas d'une fonction discriminante à deux variables, le neurone formel est composé de 2 entrées correspondant aux variables x_1 et x_2, de poids correspondant à w_1 et w_2 et d'un biais w_0.

L'entrée et les poids suivent la relation $S_j = \sum_i W_{ji} X_i$. Comme nous le verrons un peu plus loin, le neurone formel représente l'équivalent (en termes mathématiques) d'une fonction discriminante linéaire.

4.4 Discrimination linéaire à deux classes

La première notion apparaissant tout naturellement en reconnaissance de formes fut la notion de mesure de distance par rapport à une classe. Pour déterminer l'appartenance de la forme à une classe donnée, on mesurait la distance entre l'objet à classer et des classes afférentes, on classait alors l'objet dans la classe qui était la plus proche de l'objet. Cette approche était de loin la meilleure lorsque le nombre de données augmentait. Une approche

moins contraignante en temps de calcul est l'approche par les fonctions discriminantes linéaires. Celle-ci découle de la méthode par la mesure de distance et est beaucoup plus simple et plus rapide pour séparer des classes d'objets qui sont linéairement séparables, i.e pour des classes qui ne se recoupent pas (voir figure 4.9).

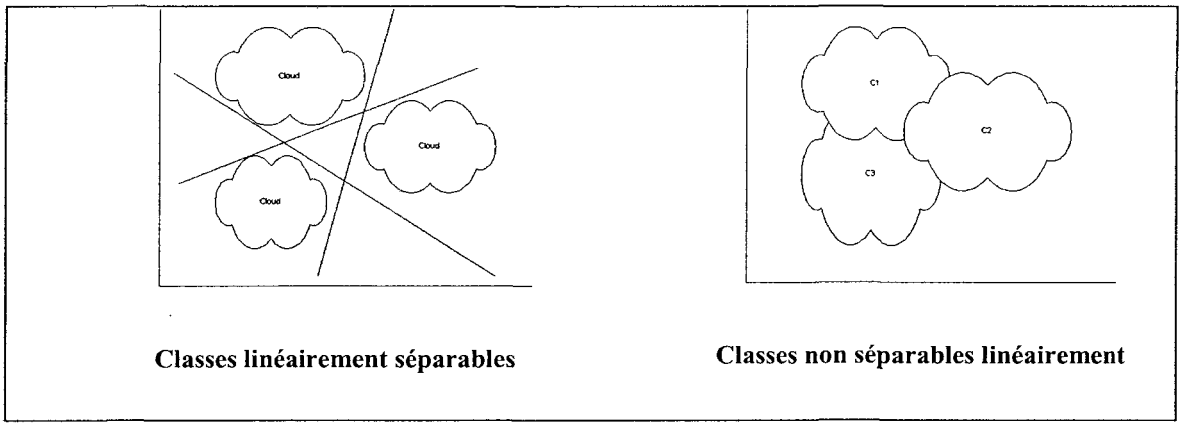


Figure 4.9 Séparation linéaire des classes

Pour comprendre le concept des fonctions discriminantes linéaires, on utilise l'approche cartésienne. Cette approche nous aidera à développer un peu plus loin le concept des réseaux neuromimétiques. Nous verrons un peu plus loin que le neurone formel découlant d'une version simplifiée du neurone biologique effectue en réalité une discrimination linéaire. Nous n'aborderons pas dans ce document l'approche biologique du neurone mais il faut savoir que le neurone formel est un modèle simplifié du fonctionnement du neurone biologique.

L'objectif de départ de la discrimination linéaire est de considérer la séparation entre deux classes par une fonction linéaire simple, soit dans ce cas-ci une droite. Dans la figure 4.10, deux classes sont séparées par une droite dont l'équation dépend de deux variables soit la variable X1 et la variable X2; ces variables étant les caractéristiques

propres à chacun des objets, caractéristiques produisant un espace à 2 dimensions dans ce cas-ci, à n dimensions dans le cas général. Dans le cas d'une forme, on choisirait par exemple la largeur et la hauteur comme caractéristiques des objets.

Pour représenter mathématiquement la classification à deux classes, nous devons représenter graphiquement une séparation de classe à partir d'une droite tracée dans un espace à deux dimensions. L'équation d'une droite est donnée par la relation suivante :

$$d(x, y) = ax + by + c = w_1 x_1 + w_2 x_2 + w_3 = 0 \qquad \textbf{Équation 4-2}$$

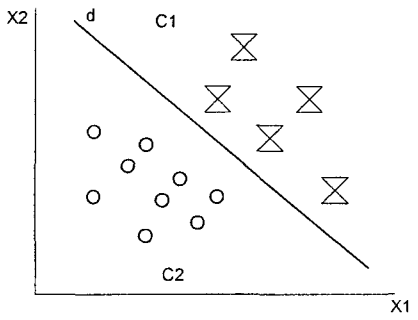


Figure 4.10 Séparation de deux classes par une droite

Cette équation est une fonction discriminante linéaire et en réarrangeant cette équation nous mettons en évidence le rôle joué par les paramètres de poids et le paramètre de biais:

$$y = \frac{-w_1}{w_2} x - \frac{w_3}{w_2} \quad \text{ou} \quad y = mx + b \qquad \textbf{Équation 4-3}$$

Nous avons remplacé x_1 et x_2 par x et y, w_1 et w_2 sont appelés des poids tandis que w_3 est appelé le biais. Les paramètres de poids et de biais sont les paramètres qui ajustent la pente et l'ordonnée à l'origine de la droite de séparation. Sous une forme plus généralisée,

nous pouvons considérer les variables de caractérisation de la forme par $\mathbf{x_i}$ et les paramètres d'ajustement de la droite de séparation par $\mathbf{w_i}$ pour donner finalement la fonction de discrimination suivante sous une forme plus compacte :

$$d(v) = w_0 x_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_i x_i = \sum_{i=0}^{n} W^T{}_i X_i = 0$$

où $W^T{}_i = (w_0, w_1, w_2, \ldots, w_i)$ et $X_i = (x_0, x_1, x_2, \ldots, x_i)$
avec $x_0 = 1$ et w_0 est le biais.

Équation 4-4

Il est évident, selon la figure 4.10, que pour d(v)>0 la donnée se classe dans C_1 tandis que pour d(v)< 0 la donnée est classée dans C_2 :

$$\begin{aligned} v &\in C_1 \quad \text{si } d(v) > 0 \\ v &\in C_2 \quad \text{si } d(v) < 0 \\ v &\in \text{Frontière} \quad \text{si } d(v) = 0 \end{aligned}$$

Équation 4-5

Ce concept se généralise pour la séparation à plusieurs classes, mais il peut arriver d'avoir des régions qui demeurent indéterminées comme on peut le voir à la figure 4.11. Aussi, dans le cas de problèmes dont les classes ne sont pas linéairement séparables, il arrive que l'on choisisse des sous-classes telles que l'on puisse les séparer convenablement.

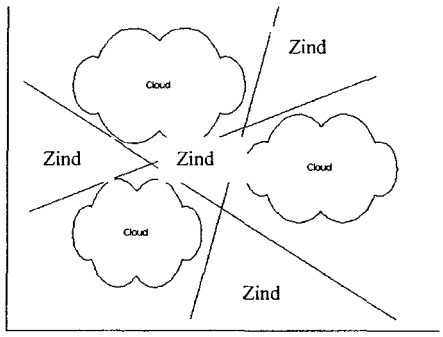


Figure 4.11 Zones indéterminées (Z_{ind}) d'un classifieur linéaire.

Dans un problème à deux classes, les paramètres de la droite, du plan ou de l'hyperplan qui sont nécessaires lors de la séparation des classes doivent pouvoir se modifier aisément et c'est là un problème qui fut résolu par les méthodes d'apprentissage.

Présentement, deux méthodes d'apprentissage sont couramment utilisées, soit l'apprentissage supervisé et l'apprentissage non-supervisé.

L'apprentissage est dit supervisé si les différentes familles des formes sont connues *a priori* et si la tâche d'apprentissage est guidée par un superviseur ou *professeur*. L'apprentissage se déroule de la manière suivante : on choisit un sous-ensemble **S** de l'ensemble **E** des formes à reconnaître. Pour chacune des formes appartenant à ce sous-ensemble, le *professeur* indique dans quelle classe doit se trouver chacune des formes. Lors de l'étape d'apprentissage, les meilleures séparatrices sont calculées pour distinguer les classes d'appartenance. Une fois complétée cette étape, on applique alors l'ensemble E des formes au modèle ainsi défini. Il est entendu que le sous-ensemble se doit d'être représentatif de toutes les données pour comporter au moins un représentant par classe. Le problème dans ce type d'apprentissage est qu'il faut un nombre élevé d'éléments dans le sous-ensemble S pour une détermination précise des droites de séparation de classes. Dans l'expérimentation, on choisit un petit ensemble de données pour chacun des objets que l'on présente en exemple au réseau de neurones. Ce sous-ensemble de données se doit d'être assez représentatif pour caractériser l'objet à être reconnu par le réseau.

L'apprentissage non-supervisé regroupe des données en formant des nuages vectoriels séparés selon une distance prédéfinie. Aucun exemple n'est nécessaire, la méthode d'apprentissage est basée sur une recherche statistique selon des caractéristiques bien précises attribuées au départ. On utilise pour ce faire soit la mesure de distance euclidienne ou soit la mesure de distance de Manhattan comme logique de séparation des nuages vectoriels. On annote à la toute fin les classes ainsi trouvées. Pour ne citer que quelques méthodes, on retrouve la méthode d'apprentissage bayésienne et la méthode GMM (Gaussian Mixture Method).

Peu importe l'algorithme d'apprentissage, il faut que les classes soient linéairement séparables, sinon on aurait des erreurs de classification. On peut citer comme exemple l'algorithme du perceptron ou l'algorithme d'Arkadev et Braverman **[42]**. L'algorithme du perceptron est plus simple. Sa convergence vers la solution est un peu plus lente mais certaine. Nous parlerons un peu plus loin du perceptron et de son utilité. Voici un exemple de calcul des poids et du biais en utilisant l'algorithme du perceptron :

Étape 1 –Choisir des valeurs aléatoires pour les poids w_1, w_2 et le biais θ

Étape 2 –Appliquer le vecteur d'entrée **x(i)**

Étape 3 -Calculer la valeur de sortie S ($S = w_1 x_1 + w_2 x_2 + \theta$)

Étape 4 – Si la valeur de sortie est égale à la valeur cible, S = d(i)

retourner à l'étape 2 sinon continuer à l'étape 5

Étape 5 –La variation du biais est égale à la valeur de la cible et la variation du poids w(i) est égale à la valeur de l'entrée x(i) fois la valeur de la cible désirée : $d\theta = \gamma d(i)$, $dw_i = \gamma x(i)d(i) -$

Étape 6 - $\theta = \theta + d\theta$, $w_i = w_i + dw_i$ et retourner à l'étape 2.

Dans l'étape 5, le symbole γ spécifie le pas d'apprentissage. Si le pas est trop petit, la convergence risque d'être faussée et si le pas est trop grand, il y a un risque de ne pas arriver à une convergence. Dans notre exemple, nous avons utilisé un pas unitaire. Après plusieurs passages de l'ensemble des vecteurs d'entrée, les poids et les biais seront ajustés de telle sorte que lors de l'application de toutes les données, celles-ci se classeront automatiquement, i.e. que la valeur de sortie S correspondra à la classe qui est associée à la donnée d'entrée.

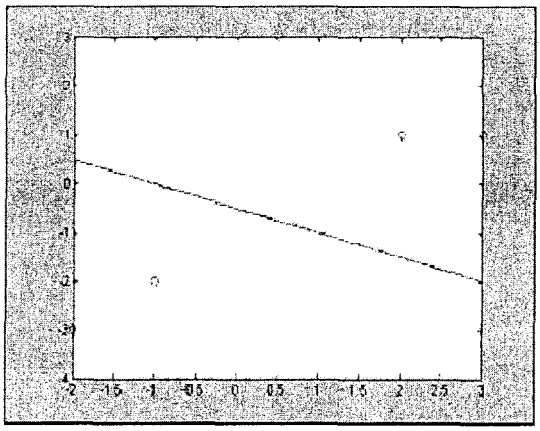


Figure 4.12 Droite séparatrice avant apprentissage.

Il faut prendre soin de remarquer que les échelles sont différentent même si les coordonnées des points demeurent identiques dans la figure 4.12 et 4.13.

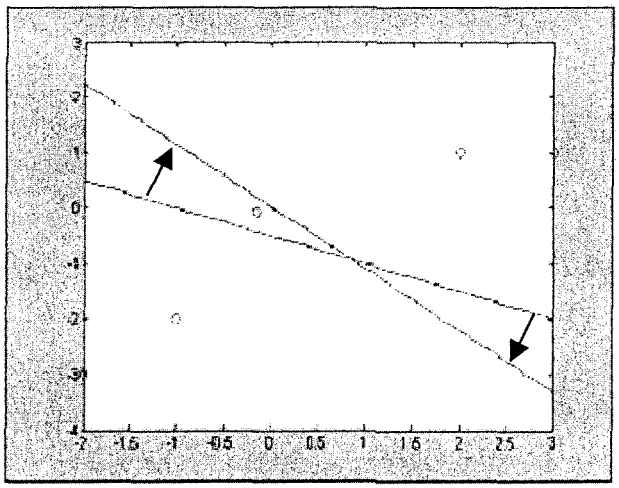


Figure 4.13 Droite séparatrice avant et après apprentissage.

Les figures 4.12 et 4.13 représentent le déplacement de la droite séparatrice avant et après apprentissage. Sur la figure 4.12, la droite sépare deux classes représentées par deux points distincts. Dans l'algorithme du perceptron, il arrive que les coefficients des poids soient à zéro parce que le taux d'apprentissage est inexistant et que la valeur des poids varie dans ce cas très fortement; il faudra dans ce cas choisir de nouvelles valeurs pour l'initialisation des poids car lorsque ceux-ci ont une valeur nulle, le calcul n'évolue plus ayant atteint un faux minimum. On doit alors reprendre l'ensemble de tous les vecteurs de la base d'apprentissage et les représenter de nouveau dans un ordre aléatoire pour donner une chance à l'algorithme de converger rapidement. La figure 4.13 présente deux droites alors que nous ne devrions voir qu'une seule droite, ceci pour indiquer l'évolution de la pente de la droite lorsqu'un nouvel objet est inséré dans la base d'apprentissage.

Voyons maintenant une autre approche qui nous est donnée par le concept de neurone formel. Le neurone formel est un modèle très simplifié simulant le fonctionnement d'un neurone biologique. Son fonctionnement s'apparente aux fonctions discriminantes

linéaires. Les neurones formels servent comme brique de base des réseaux de neurones étudiés dans ce document. Nous verrons également qu'il existe d'autres fonctions d'activation dépendant du type de réseau et des règles d'apprentissage.

Le logiciel MATLAB permet l'entraînement d'un RNA avec la fonction « **train** ». Selon les données présentées à l'entrée du réseau, la fonction « **train** » effectue le choix entre un apprentissage en mode batch ou séquentiel. Dans ce cas-ci, c'est le mode batch qui a été choisi.

Ce qui distingue un neurone formel d'une fonction discriminante linéaire est l'ajout d'une fonction d'activation qui peut être linéaire ou non. Dans le cas d'une classification binaire, la fonction d'activation utilisée est la fonction de Heaviside ou fonction seuil qui se nomme « **hardlim** » dans MATLAB. Cette fonction (figure 4.14) produit une valeur discrète égale à zéro (0) ou un (1) indiquant le résultat de la sortie. Par opposition, les fonctions discriminantes linéaires ont une fonction d'activation continue (fonction « **purelin** » dans MATLAB) voir figure 4.15.

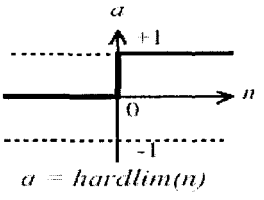


Figure 4.14 Fonction d'activation à seuil binaire

Le résultat en sortie de la fonction « **purelin** » est le même que celui présenté à l'entrée de la fonction. Cette fonction se retrouve généralement à l'entrée des réseaux

multicouches (MLP) étant donné que sa dérivée première existe. Nous verrons qu'il est important pour les réseaux multicouches d'avoir des fonctions d'activation non linéaires pour l'étape d'apprentissage car l'algorithme de rétropropagation du gradient de l'erreur doit sa fonctionnalité de la non linéarité des fonctions d'activation.

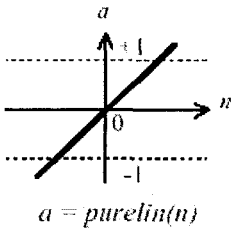


Figure 4.15 Fonction d'activation linéaire

Dans le cas où la fonction d'activation est non-linéaire, nous avons les fonctions « **logsig** » et « **tansig** ». La fonction « **logsig** » évolue entre 0 et 1 tandis que la fonction « **tansig** » évolue entre moins un (–1) et plus un (1). L'utilisation d'une fonction non-linéaire telle que le sigmoïde vient du fait que sa dérivée existe et qu'elle est simple à mettre en œuvre. Par exemple, pour la fonction « **logsig** », l'équation se traduit comme suit :

$$F(s)=\frac{1}{1+e^{-s}} \quad \text{et} \quad F'(s) = F(s)(1-F(s)) \qquad \textbf{Équation 4-6}$$

Le fait que la dérivée de cette fonction soit exprimable, de cette manière, diminue les temps de calcul pour les réseaux MLP à plusieurs neurones. Dans MATLAB, la fonction « **logsig** » a l'allure de celle présentée à la figure 4.16.

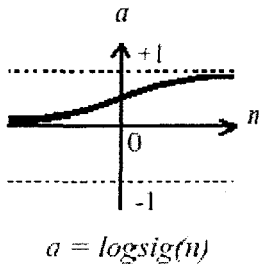


Figure 4.16 Fonction logsig

On retrouve également une autre fonction d'activation non-linéaire qui se nomme radbas et que l'on retrouve dans les réseau RBF (Radial Basis Function). Dans un réseau RBF, la première couche se compose uniquement de neurones contenant des fonctions à base radiale, celle-ci ayant l'allure d'une fonction gaussienne (voir la figure 4.17) tandis que la couche de sortie sera composée de fonctions « logsig » ou de fonctions « purelin ».

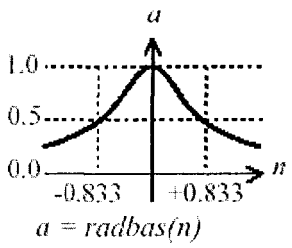


Figure 4.17 Fonction à base radiale

La fonction à base triangulaire est également disponible dans la librairie network de MATLAB. Cette fonction d'activation est rarement utilisée et sert uniquement pour les réseaux de types RBF (Radial Basis Function). On remarquera que cette fonction (figure 4.18) ressemble à la fonction Gaussienne (RadBas).

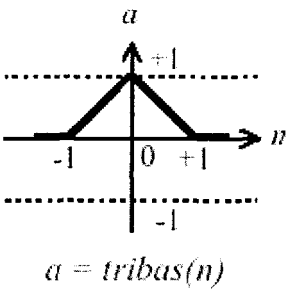


Figure 4.18 Fonction d'activation à base triangulaire

4.5 Règle d'apprentissage de Widrow-Hoff ou règle delta

Pour comprendre les réseaux multicouches, on doit obligatoirement parler de la règle de Widrow-Hoff que l'on utilise pour les réseaux à une couche de type ADALINE. Les neurones constituant ce type de réseau ont des fonctions d'activation linéaires. Cette règle d'apprentissage s'appuie sur le principe de minimisation de l'erreur quadratique instantanée entre un vecteur d'entrée **x** auquel on veut associer un vecteur de sortie Y_d .(sortie désirée). On utilise pour ce faire l'apprentissage supervisé. On se sert de cette minimisation de l'erreur quadratique pour ensuite l'appliquer, selon le principe du gradient stochastique, à la mise à jour des poids du réseau. On considère un réseau constitué de n neurones recevant des vecteurs à P composantes et dont la fonction d'activation est non linéaire. Les P entrées $\mathbf{x_i}$ du réseau sont distribuées sur tous les neurones.

La sortie du neurone i est donnée sous la forme suivante où Ω est une fonction d'activation linéaire:

$$y_i = \Omega\left(\sum_{j=0}^{P} w_{ij}x_j\right) \qquad \text{où } x_0 = 1 \qquad \textbf{Équation 4-7}$$

À un vecteur d'entrée x, on associe un vecteur de sortie $\boldsymbol{Y_{dj}}$. Faisons l'hypothèse que les poids $\boldsymbol{w_{ik}}$ ont des valeurs quelconques. Si nous comparons le vecteur de sortie désiré avec la sortie du réseau selon la valeur des poids, cette comparaison sera, a priori, non nulle; $\boldsymbol{Y_j - Y_{dj} \neq 0}$. Associons, à cette différence, l'erreur quadratique suivante :

$$E = \frac{1}{2}\sum_{j=0}^{n}\left(Y_j - Y_{dj}\right)^2 \qquad \textbf{Équation 4-8}$$

Le terme constant devant la somme permet de simplifier l'expression après dérivation. Le terme $\boldsymbol{Y_{dj}}$ est indépendant de la valeur des poids tandis que la sortie du réseau est fortement dépendante de la valeur des poids. Vérifions maintenant quelle sera la variation de cette erreur quadratique par rapport aux poids.

Soit

$$\frac{\partial E}{\partial w_{ik}} = \sum_{j=1}^{n}\left(Y_j - Y_{dj}\right)\frac{\partial\left(Y_j - Y_{dj}\right)}{\partial w_{ik}} = \sum_{j=1}^{n}\left(Y_j - Y_{dj}\right)\left(\frac{\partial Y_j}{\partial w_{ik}} - \frac{\partial Y_{dj}}{\partial w_{ik}}\right) \qquad \textbf{Équation 4-9}$$

Comme $\boldsymbol{Y_{dj}}$ est indépendant de toute variation au niveau des poids, alors

$$\frac{\partial Y_{dj}}{\partial w_{ik}} = 0 \qquad \textbf{Équation 4-10}$$

De ce fait;

$$\frac{\partial E}{\partial w_{ik}} = \sum_{j=1}^{n}\left(Y_j - Y_{dj}\right)\frac{\partial Y_j}{\partial w_{ik}} \qquad \textbf{Équation 4-11}$$

La dérivée partielle du membre de droite est annulée sauf pour j=i.

$$\frac{\partial E}{\partial w_{ik}} = (Y_i - Y_{di})\frac{\partial Y_i}{\partial w_{ik}}$$ **Équation 4-12**

Comme $y_i = \Omega\left(\sum_{j=0}^{P} w_{ij}x_j\right)$ et pour j=k, alors

$$\frac{\partial Y_i}{\partial w_{ik}} = \Omega' x_k$$ **Équation 4-13**

où Ω' est la dérivée de Ω qui est la fonction d'activation. Si nous choisissons comme fonction d'activation un sigmoïde, la dérivée de celle-ci sera donnée par la relation suivante :

$$\Omega' = \Omega(1-\Omega),$$ **Équation 4-14**

nous évitons ainsi un calcul numérique fastidieux de la dérivée de la fonction d'activation. Des équations 4.12 et 4.13 et avec $\delta_i = \Omega'(Y_i - Yd_i)$, nous aurons finalement l'équation suivante pour le gradient de l'erreur quadratique :

$$\frac{\partial E}{\partial w_{ik}} = \delta_i x_k$$ **Équation 4-15**

La mise à jour des poids s'écrira, selon le principe de descente de gradient, par

$$\Delta w_{ik} = -\alpha\delta_i x_k$$ **Équation 4-16**

Ici, le terme α est perçu comme un gain d'adaptation positif. La vitesse de convergence dépend de la grandeur de ce gain.

4.6 Le réseau MLP (Multi-Layer perceptron)

Le réseau MLP est un modèle de réseau comprenant une ou plusieurs couches cachées, dont les connexions sont directes et totales entre les couches. **Werbos**, pour sa thèse de doctorat en 1974, développa la théorie de la **rétropropagation du gradient de l'erreur**. Cet algorithme fut popularisé en 1986 par **Rumelhart** qui l'utilisa pour la modification des poids et des biais dans les réseaux de neurones multicouches. Cet algorithme modifie les paramètres de poids et de biais, en allant de la couche de sortie vers la couche d'entrée en propageant l'erreur se trouvant à la couche de sortie vers chacune des couches précédentes, l'erreur de sortie étant obtenue par comparaison de la sortie réelle z avec la sortie désirée t. L'apprentissage classique de ce réseau est conduit par une méthode de descente de gradient. Les poids sont initialisés aléatoirement et sont modifiés selon une directive qui minimise une fonction d'erreur. Cette méthode présente néanmoins un inconvénient majeur, soit celui d'une probabilité élevée de converger vers un minimum local qui faussera l'apprentissage si nous utilisons un pas d'adaptation trop petit. Nous expliquerons un peu plus loin ce qu'implique le choix de la grandeur du pas d'adaptation.

Jusqu'à maintenant, l'algorithme le plus utilisé pour l'apprentissage de ce type de réseau porte le nom de « error back propagation » ou algorithme de rétropropagation du gradient de l'erreur. La librairie « network » de MATLAB contient plusieurs variantes de l'algorithme de rétropropagation du gradient de l'erreur, la variante la plus intéressante étant l'algorithme de Levenberg-Marquardt communément appelé « faster propagation ».

4.6.1 L'algorithme de rétropagation du gradient de l'erreur

Partons de la règle de Widrow-Hoff et généralisons celle-ci pour un réseau à une couche cachée à connexions directes et totales entre les couches. À la couche d'entrée, la fonction d'activation est une fonction de type non-linéaire ou linéaire, tout dépendant des données que l'on veut présenter à la couche cachée. La fonction d'activation des couches cachées et des couches de sorties est une fonction d'activation non-linéaire. On utilise habituellement le sigmoïde. Cette fonction d'activation est très fréquemment rencontrée dans les réseaux MLP et sa dérivée est très simple (voir l'équation 4.14).

Pour les développements qui suivront, nous allons utiliser le réseau de la figure 4.19 qui représente un réseau MLP à une couche cachée.

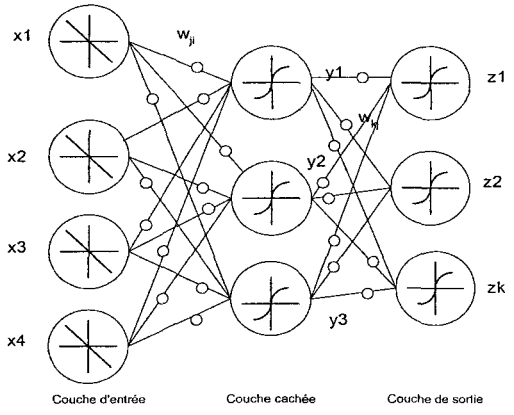


Figure 4.19 Réseau MLP à une couche cachée.

Pour le réseau à une couche cachée, les développements mathématiques seront beaucoup plus simples à comprendre. Par convention, on considère ce réseau comme ayant deux couches de traitement, soit la couche cachée et la couche de sortie, la fonction de la couche d'entrée ne faisant que distribuer les données en entrée vers la couche cachée.

Chaque unité ou neurone de la couche cachée effectue une somme pondérée de ses entrées pour former un scalaire que l'on appelle valeur d'activation du neurone. On le dénote par net_j, où j est l'indice du neurone correspondant.

$$net_j = \sum_{i=1}^{d} w_{ij}x_i + w_{0j} = \sum_{i=0}^{d} w_{ij}x_i \equiv \hat{w}_j^T \hat{x} \qquad \textbf{Équation 4-17}$$

La valeur d'activation est associée par la suite à une fonction d'activation $f(.)$ qui est une fonction linéaire facilement dérivable. Cette fonction est de la forme

$$F(net_j) = z_j = \frac{1}{1+e^{-net_j}}. \qquad \textbf{Équation 4-18}$$

On remarquera que le résultat de cette fonction se situe entre les valeurs 0 et 1. Lorsque la valeur d'activation d'un neurone est à zéro, la fonction d'activation donne la valeur 0.5, valeur qui servira d'entrée aux neurones de la couche de sortie. De part les données présentées en entrée et la sortie désirée, dans les tous premiers débuts de l'apprentissage, il existe une différence entre la sortie désirée et la sortie réelle du réseau. Pour obtenir une évolution juste de l'erreur entre la sortie désirée et la sortie réelle, nous utilisons une fonction quadratique qui agit sur tous les objets de la base d'apprentissage.

$$J(w) = \frac{1}{2}\sum_{j=1}^{n}(t_j - z_j)^2 = \frac{1}{2}\|t - z\|^2. \qquad \textbf{Équation 4-19}$$

t est le vecteur de sortie désirée contenant n composantes, z le vecteur de sortie du réseau et J, l'erreur quadratique entre la sortie désirée et la sortie réelle du réseau. w est la matrice des poids.

Le concept de base utilisé pour la rétropropagation du gradient de l'erreur est la variation de l'erreur quadratique de la couche cachée par rapport à l'erreur quadratique de la couche de sortie. Pour ajuster les poids, nous utilisons la définition qui suit, soit

$$\Delta w = -\eta \frac{\partial J}{\partial w}. \qquad \textbf{Équation 4-20}$$

On peut également écrire cette équation sous forme indicée :

$$\Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}, \qquad \textbf{Équation 4-21}$$

où η est le pas d'adaptation ou d'apprentissage du réseau. La valeur est alors mise à jour avec la relation suivante :

$$w(m+1) = w(m) + \Delta w(m) \qquad \textbf{Équation 4-22}$$

où m est l'indice associé à la présentation d'une forme particulière. Évaluons maintenant ce que représente la variation de l'erreur dans l'équation 4.22. Considérons les poids de la couche cachée vers la couche de sortie, w_{ij}.

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}} \qquad \textbf{Équation 4-23}$$

La sensibilité de l'unité k est définie comme étant

$$\delta_k = -\frac{\partial J}{\partial net_k} \qquad \textbf{Équation 4-24}$$

En présumant que la fonction d'activation est dérivable, on différentie l'équation 4.24 et on trouve que

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k}\frac{\partial z_k}{\partial net_k} = (t_k - z_k)f'(net_k). \qquad \textbf{Équation 4-25}$$

La dernière dérivée partielle dans l'équation 4.25 est déterminée en utilisant l'équation 4.18.

$$\frac{\partial net_k}{\partial w_{kj}} = y_j. \qquad \textbf{Équation 4-26}$$

La mise à jour des poids de la couche cachée vers la couche de sortie est donnée par la relation suivante :

$$\Delta w_{kj} = \eta\, \delta_k\, y_j = \eta\,(t_k - z_k)f'(net_k)\, y_j \qquad \textbf{Équation 4-27}$$

Dans le cas où les unités de sortie sont des fonctions linéaires telles que $f(net_k) = net_k$ et $f'(net_k) = 1$, alors l'équation 4-27 se ramène à la règle LMS où règle de Widrow-Hoff.

La règle d'apprentissage de la couche d'entrée vers la couche cachée est un peu plus subtile. De l'équation 4-19, en utilisant de nouveau la règle en chaîne pour les dérivées partielles, nous calculons la variation de l'erreur par rapport aux poids w_{ji}

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$ **Équation 4-28**

Développons la première dérivée, soit

$$\frac{\partial J}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2\right] = -\sum_{k=1}^{c}(t_k - z_k)\frac{\partial z_k}{\partial y_j}$$ **Équation 4-29**

La variation de la sortie par rapport à l'entrée de la couche cachée s'exprime par une nouvelle dérivée partielle qui s'écrit comme suit :

$$\frac{\partial z_k}{\partial y_j} = \frac{\partial z_k}{\partial net_k}\frac{\partial net_k}{\partial y_j} = f'(net_k)w_{kj}$$ **Équation 4-30**

Remplaçant cette équation dans l'équation 4-27, on retrouve alors la forme suivante pour $\frac{\partial J}{\partial y_j}$.

$$\frac{\partial J}{\partial y_j} = -\sum_{k=1}^{c}(t_k - z_k)f'(net_k)w_{kj}$$ **Équation 4-31**

Par analogie avec l'équation 4-23, nous utilisons l'équation 4-29 pour définir la sensibilité à l'erreur d'une unité de la couche cachée comme

$$\delta_j \equiv f'(net_j)\sum_{k=1}^{c} w_{kj}\delta k$$ **Équation 4-32**

L'équation 4-30 est le cœur de la solution concernant la connaissance de l'erreur de sortie se propageant vers les couches antérieures.

La règle d'apprentissage pour les poids de la couche d'entrée vers la couche cachée est alors donnée par

$$\Delta w_{ji} = \eta \delta_j x_i = \eta \left[\sum_{k=1}^{c} w_{kj} \delta_k \right] f'(net_j) x_i \qquad \textbf{Équation 4-33}$$

Les équations 4.27 et 4.33 forment le cœur de l'algorithme de rétropropagation du gradient de l'erreur.

4.6.2 Détection automatique d'un signal par réseau MLP

Dans cette sous-section, on démontre la possibilité pour un réseau MLP de reconnaître un signal numérique. Nous pourrions utiliser cette détection pour reconnaître des patrons contenus dans un signal donné. La réponse pourrait servir lors de l'apprentissage de signaux reliés au phytoplancton spécifique à partir d'un signal cytométrique.

À la figure 4.20, on retrouve l'enregistrement complet d'un signal quelconque contenant des patrons distincts. À partir d'échantillons de signaux contenant des formes spécifiques, on pourra apprendre à un réseau de neurones à mémoriser des patrons de signaux pour ensuite les comparer à un signal provenant d'une fenêtre échantillonnée. Les signaux de référence apparaissent sur la figure 4.21. On retrouve, dans ces diagrammes, des patrons de signaux que l'on peut retrouver dans le signal test. Pour mémoriser les signaux, on utilise un réseau MLP dont chacun des signaux sera catégorisé selon une classe spécifique.

La couche de sortie de ce réseau sera constituée d'un seul neurone dont la fonction d'activation sera une fonction linéaire, soit la fonction « purelin ». Cette fonction permet de couvrir une large bande de valeurs qui identifie distinctement les classes dont on a besoin.

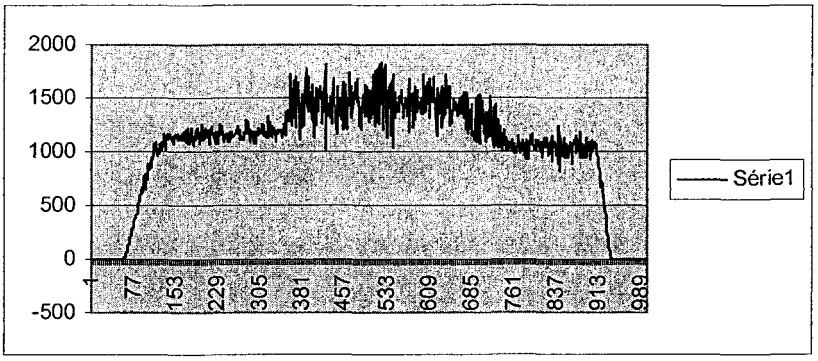


Figure 4.20 Signal de mesure

À la figure 4.21, on retrouve des échantillons de signaux correspondant à des patrons particuliers que l'on retrouve dans le signal temporel et qui seront mémorisés par un réseau neuromimétique. Les neuf patrons de signaux sont des signaux aléatoires que nous avons ajoutés dans le signal test.

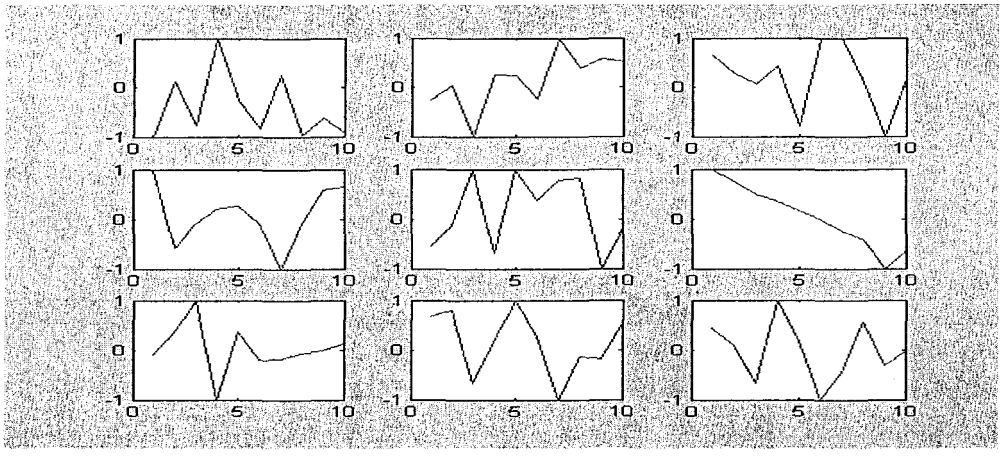


Figure 4.21 Patrons des signaux de référence

Pour mémoriser les patrons de signaux, nous utilisons un réseau MLP composé d'une couche cachée comprenant 3 neurones et une couche de sortie composée d'un seul neurone. La fonction d'activation de la couche cachée est un sigmoïde tangentiel hyperbolique dont l'équation est la suivante :

$$f(n) = \frac{2}{1+e^{-2n}} - 1 \qquad \textbf{Équation 4-34}$$

Cette fonction d'activation permet une variation de la sortie entre –1 et +1. La fonction d'activation de la couche de sortie sera une fonction linéaire « purelin » permettant d'obtenir des valeurs de sortie sans contrainte. Chacun des patrons prendra comme valeur cible celle apparaissant dans la matrice T=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9] pour catégoriser chacun des signaux. Cela veut dire que si on rencontre un patron qui fournit la valeur 0.9, alors le dernier patron est compris dans le signal de référence. Le programme permettant de comptabiliser le nombre de patrons de signaux apparaissant dans le signal de référence est présenté dans l'annexe A.

La courbe d'apprentissage de la figure 4.22 atteint rapidement la convergence désirée avec l'algorithme Levenberg-Marquardt. Cette convergence est atteinte avec seulement 3,5 itérations, ce qui est très court comparativement à d'autres méthodes d'apprentissage. Comme la quantité de patrons n'est pas élevée, le temps d'apprentissage total ne dépasse pas une seconde.

La fonction « train » permet d'entraîner le réseau à la reconnaissance des patrons tandis que la fonction « sim » simule le réseau avec la fonction de transfert déterminée lors de l'entraînement du réseau.

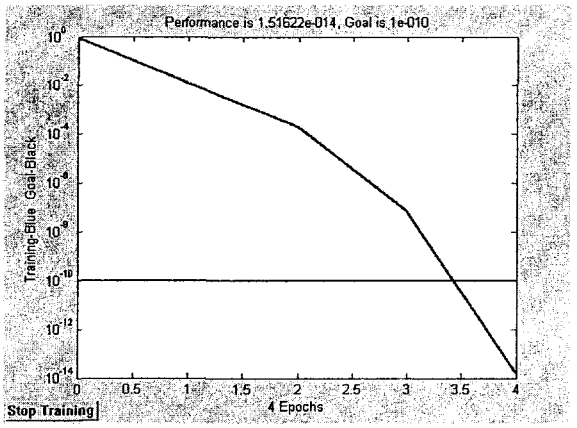


Figure 4.22 Performance du réseau MLP avec Levenberg-Marquardt.

Les résultats en sortie sont impressionnants vue le nombre de couches cachées ainsi que le nombre de neurones associés à la couche cachée. On présente ci-dessous le résultat du compte de patron rencontré dans le signal de référence.

Tdata

= 1 1 1 1 1 1 1 1 1

Chacune des colonnes de cette matrice correspond à la classe d'un patron. Les valeurs correspondent au nombre de compte de patrons enregistré dans le signal de référence. On voit que chacun des patrons se retrouve une seule fois dans le signal de référence. Comme il plane une incertitude quant à la validité des résultats, nous avons ajouté plusieurs fois le même patron (le dernier patron de signal) dans le signal de référence et avons refait une simulation pour voir si le nombre de comptes correspond au nombre de patrons se trouvant dans le signal de référence.

Le nouveau signal de référence se nomme signal_ref2 et les données se trouvent dans le fichier signal_ref2.txt. Après avoir effectué la simulation, nous obtenons les résultats suivants :

tdata =

1 1 1 1 1 1 1 1 4

Ce résultat était effectivement attendu étant donné que signal_ref2 contenait 4 signaux correspondant au dernier patron. De ce fait, il est clair que le programme effectue correctement la vérification. Dans le programme, seules les lignes faisant référence au fichier « signal_ref » ont été modifiées pour « signal_ref2 ». Il est très important de remarquer que le prétraitement des données présentées à l'entrée est une étape primordiale pour la détection des patrons. La fonction « premnmx » permet de normaliser les données pour que celles-ci puissent couvrir une échelle s'étendant entre –1 et +1.

4.6.3 Méthodes d'apprentissage pour le réseau MLP

L'apprentissage supervisé consiste à présenter au réseau des formes dont les catégories sont connues à l'avance. L'ensemble d'apprentissage est nécessaire pour permettre à un algorithme de jouer sur les poids et de fixer ceux-ci jusqu'à ce que les sorties du réseau s'approchent le plus près possible des catégories ciblées. On retrouve généralement trois protocoles d'apprentissage utilisés pour les réseaux multicouches : l'apprentissage stochastique, l'apprentissage par lot et l'apprentissage en ligne. L'apprentissage en ligne est utile lorsque l'ensemble d'apprentissage est très élevé ou que la mémoire devient prohibitive pour la sauvegarde des données; aujourd'hui, les ordinateurs

ne présentent plus de problèmes de limitation en mémoire ou de coût. Dans l'apprentissage en ligne, chacune des formes est présentée une seule fois, ce qui peut sembler donner un apprentissage très rapide mais nécessitera plus de données d'apprentissage. L'apprentissage par lot et l'apprentissage stochastique sont plus utilisés de nos jours. Dans MATLAB, nous disposons de trois protocoles d'apprentissage et nous utiliserons uniquement le protocole d'apprentissage par lot. Les méthodes vues ci-dessous proviennent toutes de la méthode de Widrow-Hoff qui est généralisée pour les réseaux multi-couches. La particularité de ces méthodes est dans la rapidité de convergence lors de l'apprentissage.

4.6.3.1 Apprentissage par lot (batch training)

La fonction « **train** » est une fonction générale effectuant l'apprentissage par lot de l'ensemble des données présentées à l'entrée d'un réseau. À partir de cette fonction, on met en marche la fonction d'entraînement du réseau qui sera associée à une fonction d'apprentissage que l'on pourra choisir selon celles indiquées ci-dessous. Dans le mode d'apprentissage par lot, les poids et les biais du réseau sont mis à jour après que l'ensemble complet des données d'apprentissage a été appliqué au réseau. Les gradients calculés après chacun des exemples d'apprentissages sont additionnés pour déterminer les changements de poids et de biais.

Exemple d'utilisation de la fonction « train » :

```
P=[-1 -1 2 2 ; 0 5 0 5] ;
T=[-1 -1 1 1] ;
Net=newff(minmax(P),[3,1],{'tansig','purelin'},'traingda') ;
[Net,TR]=train(Net,P,T);
```

Nomenclature des objets :

Net : objet identifiant le réseau.

P : matrice des vecteurs d'entrée de l'ensemble d'apprentissage.

T : matrice des vecteurs de sortie correspondant à la classe cible.

TR : objet contenant des informations sur le progrès du protocole d'apprentissage.

Newff : Fonction créant un objet réseau. Ici, on représente un réseau à une couche cachée comprenant 3 neurones et une couche de sortie contenant un neurone. Pour la couche d'entrée, P contient deux variables d'entrée et, de ce fait, les neurones de la couche cachée effectueront une somme pondérée sur deux entrées.

4.6.3.2 Descente de gradient par lot

Cette fonction est trop lente pour être utilisée dans des applications pratiques. La commande « **traingd** » est la fonction qui est utilisée pour l'apprentissage par descente de gradient par lot. Les poids sont ajustés dans la direction négative du gradient de l'erreur.

4.6.3.3 Descente de gradient par lot avec terme inertiel

Cet algorithme utilise la fonction « **traingdm** ». Cette fonction est un peu plus rapide que la fonction **traingd** mais demeure tout de même inutilisable dans les applications pratiques. Le terme inertiel agit comme un filtre passe-bas qui élimine les

variations infimes sur la surface d'erreur pour éviter l'emprisonnement dans un minimum local.

4.6.3.4 Apprentissage à pas variable

L'apprentissage à pas variable permet un compromis entre un pas trop grand amenant l'algorithme à osciller et à devenir instable tandis qu'un pas trop court amènerait une convergence plus lente et risquerait d'amener l'algorithme dans un minimum local. La fonction qui effectue ce traitement se nomme « **traingda** » pour un pas adaptatif sans terme inertiel et « **traingdx** » pour un apprentissage à pas adaptatif avec terme inertiel. Le pas d'apprentissage est responsable de la complexité de la surface d'erreur locale.

4.6.3.5 Resilient backpropagation

Habituellement, nous utilisons une fonction d'activation sigmoïde pour la couche cachée. Cette fonction est caractérisée par le fait que la pente approche la valeur zéro lorsque les données en entrée ont des valeurs très élevées. Si le gradient a une valeur très petite, cela occasionne de très petits changements dans les poids et les biais. Pour palier à ce problème, seul le signe de la dérivée est pris en compte et non la valeur de la dérivée. La fonction « **trainrp** » effectue ce genre de traitement.

4.6.3.6 Méthode d'apprentissage par algorithme du gradient conjugué

Dans cette méthode de mise à jour des poids, il existe quatre configurations possibles, soit : l'algorithme Fletcher-Reeves (**traincgf**), l'algorithme Polak-Ribiére (**traincgp**), l'algorithme de redémarrage Powell-Beale (**traincgb**) et enfin l'algorithme du conjugué du gradient d'échelle (**trainscg**).

4.6.4 Les algorithmes Quasi-Newton

Les algorithmes basés sur la méthode de Newton sont une alternative aux méthodes du conjugué du gradient pour atteindre rapidement l'optimisation des paramètres de poids et de biais. La méthode de Newton se base sur le calcul suivant

$$x_{k+1} = x_k - A_k^{-1} g_k \qquad \textbf{Équation 4-35}$$

où **A** est la matrice hessienne (matrice des dérivées secondes de l'indice de performance des valeurs de poids et de biais). Les algorithmes provenant de la méthode de Newton convergent beaucoup plus rapidement que les méthodes du conjugué du gradient. Malheureusement, la matrice hessienne consomme un temps appréciable en calcul. Il existe des méthodes que l'on appelle quasi-newton où le calcul de la matrice hessienne est évalué approximativement à chaque pas d'itération de l'algorithme. On retrouve ces algorithmes dans la fonction « **trainbfg** ». Dans cette fonction, on retrouve d'autres méthodes de convergence dont l'algorithme de sécante « trainoss », l'algorithme Levenberg-Marquardt « trainlm » et l'algorithme de Levenberg-Marquardt à mémoire réduite « trainlm ». Ceci passe en revue les protocoles d'apprentissage pour les réseaux multicouches. Dans l'expérimentation, nous utilisons la méthode de Levenberg-Marquardt car elle est la plus rapide. Le seul défaut de cette méthode est de pouvoir être appliquée à un réseau contenant moins d'une centaine de poids à mettre à jour, donc un réseau de neurones qui doit être optimisé pour pouvoir utiliser cette méthode de convergence.

4.7 Les réseaux à apprentissage compétitif

Les réseaux à apprentissage compétitif apprennent à grouper des vecteurs d'entrées similaires de telle sorte que ces réseaux trouvent leur utilité dans la détection de régularités et de corrélation dans les données présentées à leur entrée. On dispose de plusieurs méthodes d'apprentissage compétitif telles que les cartes auto-organisatrices avec la méthode d'apprentissage de Kohonen (SOM), les apprentissages VQ et LVQ. Le réseau SOM est un réseau compétitif qui apprend à catégoriser les vecteurs d'entrée qui lui sont présentés. Dans le toolbox de MATLAB, on retrouve deux types de réseaux : les réseaux compétitifs (créés par la fonction newc) et les réseaux auto-organisateurs (créés par la fonction newsom). On retrouve pour les réseaux compétitifs différentes méthodes d'apprentissage, la plus courante étant la méthode LVQ. La méthode LVQ est une méthode d'apprentissage supervisé pour catégoriser les vecteurs d'entrée contrairement au réseau SOM qui catégorise les vecteurs d'entrée selon une mesure de distance, ce qui peut entraîner un mauvais classement si deux vecteurs sont très similaires. La méthode VQ utilise une méthode d'apprentissage non supervisée comme nous le verrons plus loin.

4.7.1 VQ: Vector Quantization (quantification vectorielle).

Introduite par Grossberg (1976), la quantification vectorielle est une méthode généralement qualifiée d'estimateur de densité non supervisé (apprentissage non supervisé).

Elle permet de retrouver des groupes sur un ensemble de données, de façon relativement similaire à un algorithme de type *k-means* que l'on préfèrera d'ailleurs généralement à un VQ si la simplicité d'implémentation n'est pas un élément majeur de la résolution du problème. Cette méthode de classification était très utilisée récemment mais aujourd'hui on lui préfère la méthode LVQ car celle-ci emploie une technique d'apprentissage supervisé tandis que la méthode VQ emploie une technique d'apprentissage non-supervisé.

4.7.2 Les réseaux SOM (Self Organizing Map).

Les réseaux SOM sont issus des travaux de **Fausett** (1994) et **Kohonen** (1995). Ces réseaux sont très utilisés pour l'analyse de données. Ils permettent de cartographier en deux dimensions et de distinguer des groupes dans des ensembles de données. Les SOM sont encore largement utilisés mais les scientifiques leur préfèrent maintenant les LVQ (Learning Vector Quantization). Les réseaux de type SOM utilisent une méthode d'apprentissage non supervisée tandis que le réseau LVQ utilise une méthode d'apprentissage supervisée. La création d'un réseau SOM s'effectue à l'aide de la fonction « newsom ». Le réseau de Kohonen agit comme un outil de compression de données si on l'applique à des données préalablement traitées avant présentation à un réseau de neurones de type MLP. Nous allons montrer comment un réseau SOM permet la compression de données à l'aide d'un exemple.

Le réseau SOM compétitif utilisé dans cet exemple est composé de 50 neurones dont les poids s'organisent pour former une carte auto-organisatrice des vecteurs d'entrée. Le SOM permet un prétraitement des données tout comme le fait la méthode PCA. Le SOM

quantifie les vecteurs principaux tout en éliminant les autres vecteurs qui sont fortement corrélés. Dans cet exemple de réseau SOM, le nombre d'itérations et le pas de quantification de Kohonen sont choisit de façon arbitraire et ensuite optimisée selon les résultats obtenus. Voici ce que donne le réseau SOM lorsque nous effectuons l'apprentissage non supervisé pour un nombre d'itération égal à un. On retrouvera en annexe le code MATLAB permettant un apprentissage non supervisé (« clustering ») d'un nuage de points par un réseau SOM.

Les données présentées à l'entrée de ce réseau ont été choisies de telle sorte que nous puissions « voir » la topologie résultante qui donne une forme plus allégée dont le résultat pourra ensuite être transmis à un réseau MLP. On voit que pour un nombre relativement faible de neurones (50 neurones), la forme topologique conservée est quelque peu grossière pour la quantité de données (1000 vecteurs) présentée à l'entrée du réseau comme on peut le voir à la figure 4.23.

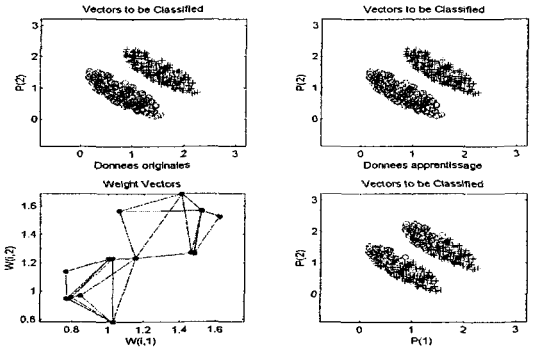


Figure 4.23 Réseau de Kohonen de 10x5 neurones

Par contre, si nous augmentons le nombre de neurones du réseau de 50 à 2500 (figure 4.24), nous obtenons un meilleur résultat au niveau topologique. Il faut voir dans la

topologie une forme de segmentation qui réduit considérablement le nombre de pixels d'une image qui serviront comme paramètres d'entrée à un réseau MLP.

Pour un nombre d'itérations plus élevé, on aura une meilleure topologie mais un temps de calcul plus long. Il s'agira de faire un compromis entre temps de calcul et nombre de neurones.

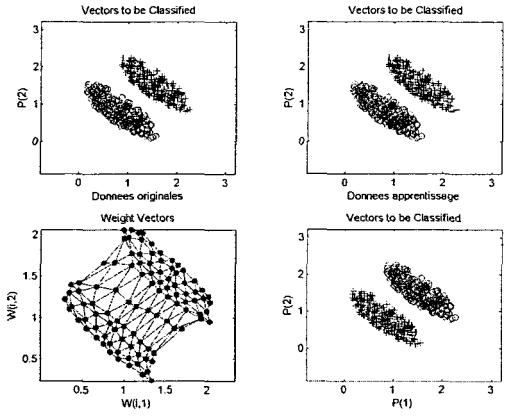


Figure 4.24 Carte auto-organisatrice de dimension 50x50

On remarque que la topologie est conservée dans la forme de l'objet présentée au réseau. Si on choisit un nombre de neurones plus petit que le nombre de vecteurs présentés à l'entrée, la topologie est encore conservée et les données représentées par les poids constituant le réseau peuvent alors servir comme nouvelle entrée pour un réseau MLP. Il faut que le nombre d'itérations soit assez élevé pour que la carte auto-organisatrice puisse approcher la topologie des vecteurs d'entrée.

Le réseau LVQ donne une meilleure topologie avec un nombre de neurones moindre que pour le réseau SOM. Ceci est dû en grande partie à la méthode d'apprentissage

supervisé dans le cas du réseau LVQ contrairement à un apprentissage non-supervisé dans le cas du réseau SOM.

4.7.3 LVQ: Learning Vector Quantization.

Les réseaux utilisant la méthode LVQ ont été proposés par Kohonen (1988). Des trois types de réseaux présentés ici, cette méthode est la seule qui soit réellement adaptée à la classification de données par "recherche du plus proche voisin". La méthode d'apprentissage est supervisée. Le logiciel MATLAB fournit deux fonctions de type LVQ, soient « learnlv1 » et « learnlv2 ».

4.8 Les réseaux à fonctions noyaux (*réseaux de type bayésien*)

Les réseaux RBF (Radial Basis Function) sont des réseaux à couches dont l'origine remonte à une technique d'interpolation RBF. Broomhead et Lowe ont été les premiers à utiliser cette technique dans les réseaux de neurones. Le réseau RBF est constitué d'une couche cachée dont les fonctions d'activation ont l'allure d'une courbe gaussienne. La couche de sortie est la plupart du temps constituée de fonctions d'activation linéaires. La représentation schématique de ce type de réseau est donnée à la figure 4.25:

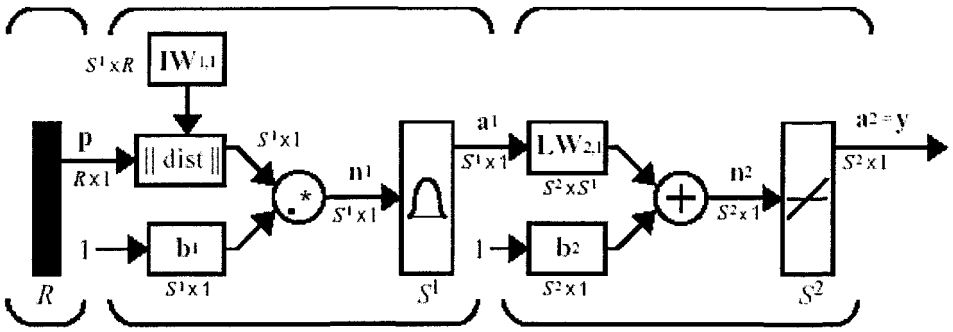


Figure 4.25 Schéma d'un réseau RBF

L'apprentissage des réseaux RBF est généralement composé de deux étapes distinctes : la paramétrisation des fonctions-noyaux et l'apprentissage des poids de la couche de sortie. Cette dernière étape ne concerne qu'une seule couche de connexions et peut être réalisée par des techniques d'apprentissage simples telles que la règle Delta ou de Widrow-Hoff. Les réseaux RBF sont capables de calculs très puissants. Avec une seule couche cachée, les réseaux RBF se comportent comme des approximateurs universelles.

Les réseaux RBF obtiennent des performances comparables ou supérieures à celles des réseaux MLP dans l'approximation de fonctions. Une particularité intéressante est la rapidité et la simplicité dans l'apprentissage faisant de ces réseaux des outils de choix pour des systèmes de classification de signaux. D'un point de vue pratique, les réseaux RBF sont moins sensibles aux pertes de mémoire résultant de la destruction de leurs poids.

Dans la configuration de base, ce type de réseau est constitué de deux couches dont une agit comme la couche cachée. La première couche qui n'est pas comptabilisée est la couche d'entrée dont les fonctions d'activation sont linéaires, telle que la fonction « purelin ». La couche cachée est constituée de fonctions de type gaussien telle que la fonction « radbas » fournie par MATLAB. Finalement, la couche de sortie comporte des fonctions d'activation non-linéaires telles que la fonction « tansig ». On utilise une méthode d'apprentissage supervisé pour ce type de réseaux. On pourrait également utiliser une méthode d'apprentissage non supervisé de type VQ mais cela demanderait énormément de données pour l'obtention de résultats concluants.

4.8.1 Approximation d'une fonction par un réseau Bayésien.

Un exemple typique d'application des réseaux Bayésien consiste à approcher la forme d'onde d'une fonction quelconque. On peut voir à la figure 4.26 que pour un vecteur d'apprentissage contenant 21 données discrétisant une forme d'onde, le réseau RBF parvient à reconstituer cette même forme d'onde dans le domaine continu avec une très bonne approximation. S'il exite un trop grand nombre de neurones dans la couche cachée, la fonction peut être surévaluée, c'est-à-dire que tous les points appartiennent à la fonction évaluée, sauf qu'entre les points, nous obtenons des valeurs qui ne correspondent plus à la fonction; la fonction subissant des oscillations entre ces points dû à un trop grand nombre de degrés de liberté dans le réseau. Il ne faut pas perdre de vue que la généralisation du réseau est une caractéristique importante dans la reconnaissance des fonctions. Une fonction dont les données sont bruitées se doit d'être reconnue de par la généralisation du réseau. Si le réseau n'arrive pas à généraliser convenablement, le signal ne sera pas reconnu. Donc il ne faut pas que le réseau soit surévalué pour obtenir une bonne généralisation.

Sur le graphe 1 de la figure 4.26, on retrouve les vecteurs d'apprentissage qui servent dans l'approximation d'une fonction pour le réseau RBF. Le graphe 2 donne la forme de la fonction d'activation radiale, soit ici une gaussienne, le graphe 3 donne les gaussienne qui se présentent à l'entrée de la couche de sortie. Celles-ci seront sommées par les neurones de la couche de sortie et enfin le graphe 4 présente le résultat de l'approximation de la fonction.

L'expérimentation consiste à soumettre l'entrée d'un RNA les trois vecteurs de paramètres représentant les caractéristiques d'un système. La fonction résultante est une forme d'onde que le réseau doit apprendre à approximer. Chacun des vecteurs contient 51 valeurs. On présente en annexe B six (6) graphiques auquels nous avons ajouté un bruit différent pour chacun des six (6) signaux.

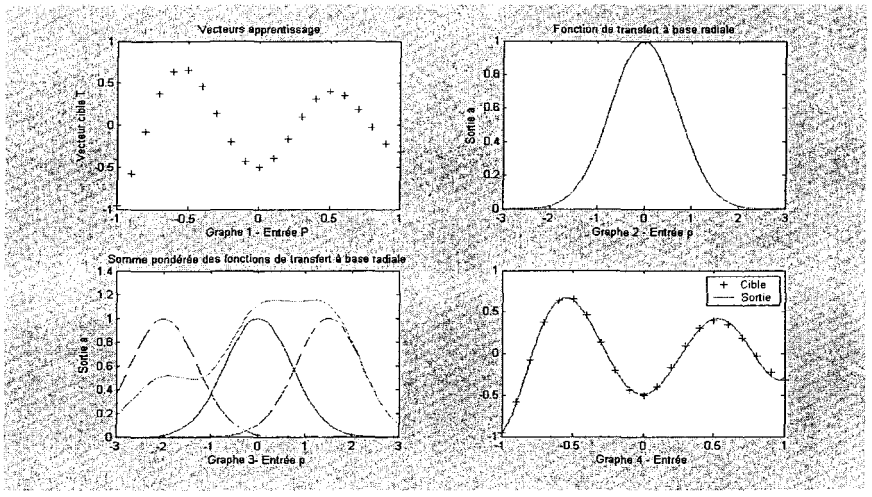


Figure 4.26 Approximation d'une fonction par un réseau RBF

Le programme permettant l'expérimentation d'un réseau RBF est présenté en annexe.

4.9 Les réseaux récurrents

Les réseaux récurrents sont considérés comme d'un intérêt considérable dans la recherche actuelle. On en trouve deux types, soit les réseaux de Elman et le réseau de Hopfield. Les réseaux de Elman permettent de reconnaître et de générer des patrons de signaux temporels et spatiaux tandis que le réseau de Hopfield agit comme correcteur d'erreur ou réseau à catégorisation vectorielle.

L'utilisation des réseaux récurrents apparaît aujourd'hui comme une autre alternative pleine de promesse. Des recherches sont en cours actuellement pour comprendre comment interagissent ces types de réseau selon les applications considérées.

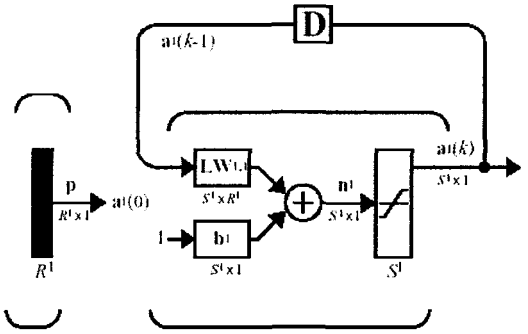


Figure 4.27 Réseau de Hopfield

Ces réseaux sont un peu plus complexes que les perceptrons multicouches. Chaque cellule est connectée à toutes les autres et les changements de valeurs des cellules s'enchaînent en cascade jusqu'à un état stable. Les réseaux récurrents sont bien adaptés à la reconnaissance de formes. Les réseaux de Hopfield sont en évolution constante mais pas assez pour leur trouver des applications pratiques fonctionnelles. Leur curiosité est plus d'ordre théorique que pratique.

4.10 La méthode Taguchi dans l'optimisation des réseaux de neurones

Dans la gestion de la qualité ou des procédés, il est de nature courante d'établir un protocole expérimental permettant de déterminer les causes ou facteurs pouvant influencer la qualité d'un produit ou d'un procédé. Le résultat qui découle de la variabilité de ces facteurs est appellé la réponse du système. La planification des essais vise à produire le maximum de résultats pour une dépense aussi faible que possible. La première méthode qui nous vient à l'esprit est de faire varier un seul facteur en laissant fixe les autres, permettant ainsi de connaître l'effet sur la réponse: cette méthode a cependant le désavantage d'être couteuse en temps et en matériel.

Il n'y a pas de méthode générale d'optimisation de processus et s'il existe des interactions entre les facteurs, ce que l'on nomme couplage de paramètres, on ne pourra déterminer que difficilement un modèle prédictif. Dans un plan d'expériences, toutes les données sont employées simultanément pour calculer chacun des effets amenant un plus petit nombre d'essais et une meilleure précision. Le résultat de l'essai pourra découler d'une expérience matérielle ou d'une simulation numérique. C'est ce que la méthode Taguchi permet de réaliser **[31]**. Cette méthode est issue des travaux de Fisher (1925) et remise en vogue dans les années soixante par le Dr. Taguchi. La planification des expériences a connu, ces dernières années, de nombreuses innovations dans des secteurs aussi variés que les réseaux de neurones et la commande moderne. Le plan d'expériences sera utilisé lors de l'expérimentation pour trouver une configuration optimale du réseau de neurones. Le lecteur pourra consulter l'annexe pour un aperçu de cette technique.

4.11 Conclusion

Nous avons décrit les fondements des réseaux de neurones artificiels à partir de la formulation mathématique d'un neurone formel. On a vu également que l'entraînement des réseaux de neurones s'apparente plus à un art qu'à une science de même que la détermination de la meilleure configuration.

Les expériences réalisées dans ce document ont permis une meilleure compréhension des RNA ainsi que de leurs limitations. Nous avons vu plusieurs configurations de réseaux ainsi que plusieurs modèles d'apprentissage et nous en avons fait la comparaison du point de vue de la vitesse d'apprentissage et des résultats fournis.

Il appert que le réseau RBF est l'une des meilleures configurations pour l'approximation de fonctions apportant une configuration optimale parce que nous avions affaire à un réseau évolutif tandis que pour le réseau MLP, il est difficile de plancher sur la meilleure configuration étant donné le caractère stochastique du nombre de couches nécessaires et du nombre de neurones nécessaire par couches permettant une configuration optimale. La méthode des plans d'expériences présente une évolution dans la recherche d'une configuration optimale sur le processus d'apprentissage vs le processus de reconnaissance par le choix du nombre de couches et du nombre de neurones par couches.

Nous avons vu également les avantages et désavantages des types d'apprentissage, soit l'apprentissage supervisé ou l'apprentissage non-supervisé, chacun ayant ses caractéristiques propres.

L'apprentissage supervisé peut être long et fastidieux selon certaines applications mais apporte tout de même une plus grande précision quant à la définition des classes, uniquement si les données cibles sont très représentatives de l'ensemble des données. Le réseau MLP comparativement au réseau compétitif (réseau de Kohonen) apporte des résultats très significatifs au niveau de la reconnaissance de formes, le taux de précision de la classification étant meilleur autant pour l'un que pour l'autre. Il est plus qu'avantageux de faire apprendre à un réseau un comportement bruité qu'un comportement précis parce que la généralisation est conservée contrairement à un apprentissage trop fin qui ferait perdre toute généralisation du réseau.

Finalement, pour conclure, nous en sommes encore à l'étape des essais et erreurs quant à la meilleure configuration que nous devons donner à un réseau de neurones pour résoudre de manière optimale un problème spécifique. Il est à espérer que dans un proche avenir les développements dans ce champ de recherche permettront de mieux saisir toute la complexité du cerveau humain et par le fait même nous permettre de découvrir de nouvelles possibilités d'utilisation des réseaux de neurones artificielles.

CHAPITRE 5

MÉTHODOLOGIE ET MANIPULATIONS

5.1 Introduction

Nous disposons de trois sources de données contenant plusieurs centaines d'images ainsi que les paramètres cytométriques correspondants. La première source de données annotée base de données #1 contient 3279 images de cellules ne faisant pas partie de l'espèce que nous voulons détecter. Certaines images de la base de données #1 contiennent des cellules apparentées à *Alexandrium tamarense* qui ne contiennent aucune toxine dangereuse pour l'être humain. La seconde source de données, qui est annotée base de données #2, contient 183 images composées uniquement de cellules d'*Alexandrium tamarense* qui serviront pour l'étape d'apprentissage du modèle. En ce qui concerne la troisième source de données, annotée base de données #3, celle-ci inclut quelques images de cellules *d'Alexandrium tamarense* parmi un ensemble d'images disparates de cellules qui sont ou non apparentées à l'espèce *Alexandrium tamarense.*

L'expérimentation s'effectue sur deux stratégies dans ce travail de recherche. Nous avons utilisé cette méthodologie comme une voie de comparaison entre la technique d'extraction de paramètres par traitement d'image avec celle utilisant uniquement les paramètres cytométriques obtenus par le FLOWCAM.

La stratégie un (1) dont les images proviennent du FLOWCAM sont traitées par la méthode des moments d'ordre n avec n<3 dont les paramètres sont fusionnés avec quatre paramètres cytométriques de base que l'on présente à un réseau de neurones pour fin de classification. L'ensemble des paramètres est par la suite décorrélé pour ensuite être présenté à un réseau de neurones.

La stratégie deux (2) utilise tous les paramètres générés par le FLOWCAM avec et sans utilisation de la technique de décorrélation. Les étapes de normalisation sont appliquées aux données avant l'application de la méthode de décorrélation et ensuite dénormalisé pour être appliqué directement à un réseau de neurones.

Dans les deux stratégies, on utilise le même procédé d'apprentissage avec 33% des données obtenues aléatoirement selon une loi normale. Le mode d'apprentissage que nous préconisons pour l'ensemble de ces données est le mode d'apprentissage supervisé à cause du faible nombre de données disponible pour l'expérimentation. Le type de réseau que l'on utilise dans cette expérimentation est un réseau MLP avec calcul des poids du réseau qui s'effectue par rétropropagation du gradient de l'erreur. Nous avons expérimenté avec ce type de réseau car, à la lecture des documents concernant les réseaux de neurones, il est fait souvent mention de la place qu'occupe ce type de réseau dans les applications d'ingénierie.

5.2 L'approche expérimentale

5.2.1 Les objectifs de recherche

L'objectif de cette recherche est de détecter et reconnaitre une espèce particulière de phytoplancton (*Alexandrium tamarense*) parmi des espèces apparentées ne présentant pas la toxicité d'*Alexandrium tamarense*. Certaines recherches dont la classification d'image de phytoplancton était uniquement basée sur la reconnaissance de formes **[23]** ont démontré que c'est une voie de recherche qui présente peu de résultats concluants; la principale difficulté se trouve dans la recherche d'invariants qui caractérisent de manière unique une forme déterminée pour des images en deux dimensions. L'approche préconisée dans cette recherche est d'associer des paramètres cytométriques aux paramètres obtenus par le traitement des images. Nous vérifions si l'ajout de paramètres cytométriques peut améliorer la classification afin d'éviter que des espèces apparentées puissent être classées dans la même catégorie que l'espèce présentant une forte toxicité.

5.2.2 Hypothèses

On considère que les données présentées à l'entrée du réseau de neurones lors de l'étape d'apprentissage forment une distribution suivant une loi normale. Cette hypothèse est utilisée lors de la présentation des données d'apprentissage au réseau de neurones, dont seulement 33% sont choisies aléatoirement.

5.2.3 Description des données

Pour chacun des échantillons analysés par le logiciel intégré du FLOWCAM, le logiciel de celui-ci génère plusieurs fichiers dont des images avec extension .tif (voir figure 5.1), des fichiers contenant des paramètres avec extension .fcm (voir figure 5.3) dont les informations spécifiques pour chacune des cellules sont composées de paramètres liés tant à l'image qu'au cytomètre. Ces paramètres indiquent le nombre de cellules apparaissant dans l'image, la longueur et la largeur de la cellule, le centre de gravité, la fluorescence engendrée par l'excitation du pigment de la chlorophylle et de la phycoérythrine par un ou des lasers ainsi que d'autres paramètres pouvant caractériser une cellule par rapport à d'autres.

Par cette description des données, nous voulons nous assurer que le lecteur aura une compréhension globale des processus mis en jeu dans cette recherche. La qualité des images produites par le FLOWCAM joue énormément sur le calcul des paramètres et si l'image est très bruitée, les valeurs des paramètres seront faussées et une mauvaise classification des données s'en suivra.

La figure 5.2 présente une planche d'images contenant uniquement des cellules d'*Alexandrium tamarense*. Cette planche fait partie d'un groupe de six (6) planches images dont les cellules ont des formes et des données cytométriques variées. Une partie de ces images serviront à l'étape d'apprentissage du réseau de neurones.

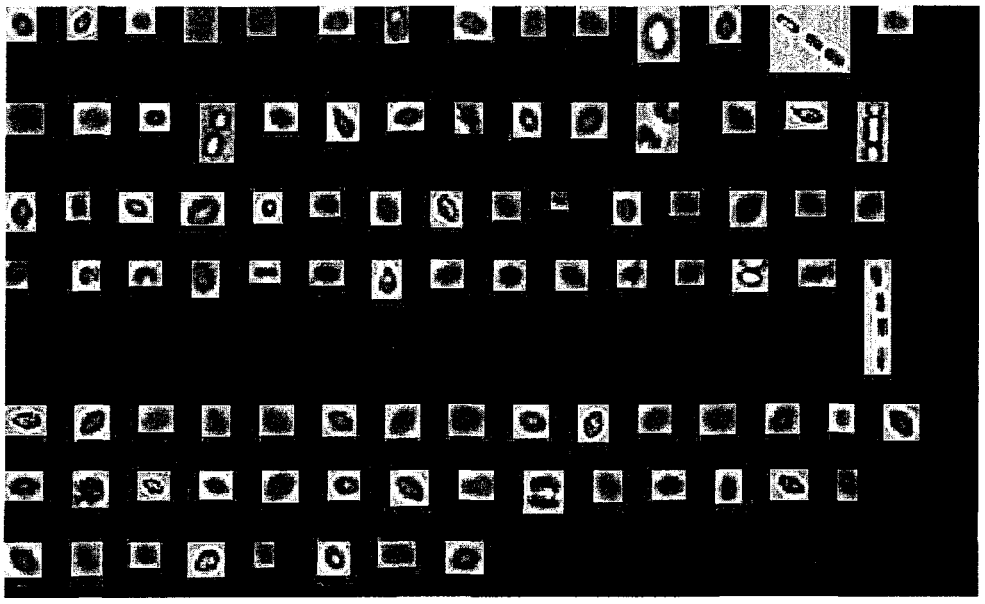


Figure 5.1 Exemples d'échantillon d'images du FLOWCAM

Département de biologie de l'Université Laval

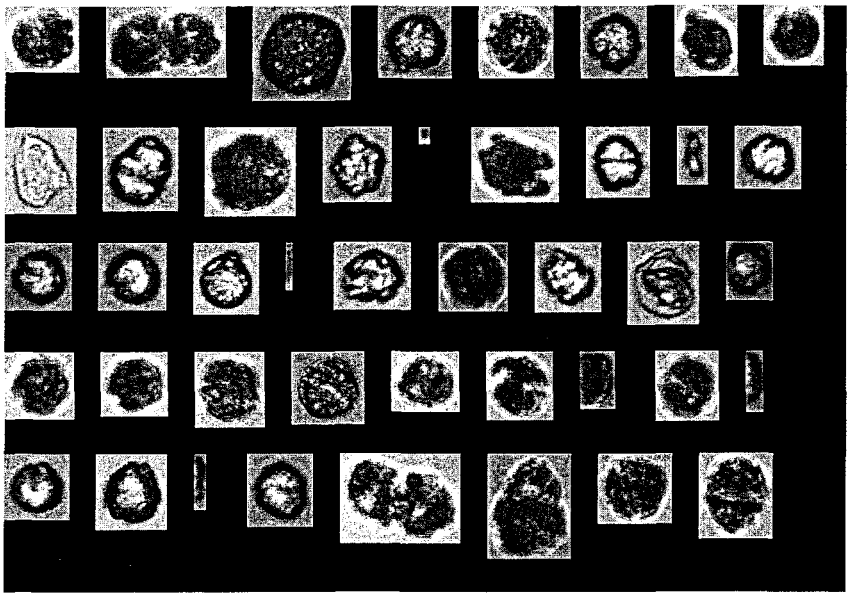


Figure 5.2 Planche d'images d'Alexandrium tamarense

Département de biologie de l'Université Laval

Time of Event	File Number	Cell Area	Chlorophyll Peak	Chlorophyll TOF	Particles per chain	FFT Area	Phyco Peak	Phyco TOF	feret_max_diameter	feret_min_diameter	Cell X	Cell Y	Cell ESD	Pat. rec. score	FlowCam Vote	Manual Vote(-1 => no entry)	Stuck Particle Flag (1=> stuck)	histogram bins/freq	Built Fluor (uVolts)	
170057617	170135235	25	2289	5430	1	2	2264	7970	8	6	103	234	5	0	0	-1	0	0	0	0 2289
170057617	170135235	26	2362	5586	1	11	2239	7498	8	5	342	373	5	0	0	-1	0	0	0	0 2362
170057617	170135235	24	2385	5457	1	25	2376	7319	6	4	179	16	5	0	0	-1	0	0	0	0 2385
170057617	170135235	47	2457	6979	1	4	2373	7290	9	8	492	89	7	0	0	-1	0	0	0	0 2457
170057617	170135235	73	2246	6923	1	2	2240	7113	13	9	462	405	9	0	0	-1	0	0	0	0 2246
170057617	170135235	26	2264	7256	1	3	2244	7440	9	5	442	254	5	0	0	-1	0	0	0	0 2264
170057617	170135235	55	2355	7662	1	12	2342	7797	13	10	364	395	8	0	0	-1	0	0	0	0 2355
170057617	170135235	38	2256	2659	1	3	2253	2658	10	7	187	255	6	0	0	-1	0	0	0	0 2256
170057617	170135235	19	2267	1986	1	18	2266	2035	7	4	473	191	4	0	0	-1	0	0	0	0 2267
170057617	170135235	26	2350	7847	1	3	2316	7666	9	5	591	338	5	0	0	-1	0	7	0	0 2350
170057617	170135235	78	2285	6800	1	4	2272	7311	18	12	289	32	9	0	0	-1	0	42	0	0 2285
170057617	170135235	49	2363	6817	1	7	2254	7422	10	7	517	186	7	0	0	-1	0	47	0	0 2363
170057617	170135235	74	2495	12866	3	8	2269	7253	27	4	265	17	9	0	0	-1	0	42	0	0 2495
170057617	170135235	33	2352	5802	1	4	2281	1158	8	5	261	221	6	0	0	-1	0	24	0	0 2352
170057617	170135235	44	2352	6802	1	4	2281	1158	10	7	209	283	7	0	0	-1	0	40	0	0 2352
170057617	170135235	39	2535	852	1	6	2241	1230	9	7	262	105	7	0	0	-1	0	17	0	0 2535
170057617	170135235	27	2535	852	1	6	2241	1230	7	6	143	355	5	0	0	-1	0	4	0	0 2535
170057617	170135235	46	2584	6886	2	27	2265	7846	19	7	7	42	7	0	0	-1	0	3	0	0 2584
170057617	170135235	33	2439	7444	1	12	2265	8156	8	5	136	300	6	0	0	-1	0	4	0	0 2439
170057617	170135235	43	2433	1292	1	3	2295	7552	12	6	322	252	7	0	0	-1	0	7	0	0 2433
170057617	170135235	42	2433	1292	1	3	2295	7552	9	6	373	336	7	0	0	-1	0	1	0	0 2433
170057617	170135235	27	2424	7286	1	5	2273	7746	8	5	465	52	5	0	0	-1	0	0	0	0 2424
170057617	170135235	30	2491	4984	1	5	2257	7944	9	5	402	10	6	0	0	-1	0	0	0	0 2491
170057617	170135235	38	2420	7166	1	8	2286	8133	11	7	519	247	6	0	0	-1	0	0	0	0 2420
170057617	170135235	79	2537	6554	2	8	2294	431	21	6	399	366	10	0	0	-1	0	0	0	0 2537

Figure 5.3 Exemple de fichier généré par le logiciel associé au FLOWCAM

Les paramètres se trouvant dans le fichier généré par le FLOWCAM apparaissent dans le tableau 5-1 : ce sont les paramètres de base que nous avons choisi pour la stratégie deux (2). Pour avoir accès à un plus grand nombre de paramètres, nous construisons par diverses combinaison des paramètres supplémentaires que nous passerons ensuite dans un décorrélateur utilisant la méthode PCA (Analyse par composantes principales).

Cell area
Chlorophyll peak
Chlorophyll TOF
particles per chain
FFT Area
Phyco Peak
Phyco TOF
Feret_max_diameter
Feret_min_diameter
Cellx
Celly
Cell ESD

Tableau 5-1 Paramètres du fichier fcm du FLOWCAM

Nous voulons obtenir, par cette manipulation, la possibilité d'obtenir plus de paramètres orthogonaux après le passage dans un décorrélateur. Les paramètres « Cell Area », « FFT Area », « Feret_max_diameter », « Feret_min_diameter », « Cellx », « Celly » et « Cell ESD » sont liés à la forme de la cellule. Ces paramètres de formes sont le résultat d'un traitement d'image effectué par le FLOWCAM. Le paramètre « Chlorophyl peak » correspond à la valeur maximale de fluorescence de la présence de pigment de chlorophyle enregistrée lors du passage de la cellule devant les capteurs. Le paramètre « Chlorophyl TOF » désigne le temps de passage en milliseconde du signal de fluorescence de la chlorophyle. Le paramètre « Phyco peak » est la valeur maximale de fluorescence obtenue lors de l'excitation par un laser de longueur d'onde spécifique du pigment de phycoérythrine contenu dans une cellule tandis que le paramètre « Phyco TOF » est le temps de passage en milliseconde du signal de fluorescence de la phycoérythrine. Ces paramètres proviennent de la partie cytométrique du FLOWCAM.

Pour la stratégie un (1), nous avons utilisé uniquement les paramètres cytométriques représentés par les colonnes « Chlorophyl peak », « Chlorophyl TOF », « Phyco peak » et « Phyco TOF » tandis que pour la stratégie deux (2), nous avons utilisé tous les paramètres du tableau 5-1. Dans la stratégie un (1), chaque groupe de paramètres, pour une cellule distincte, est fusionné avec ceux calculés par les moments d'ordre $n<3$ pour chaque image de cellules.

Dans le tableau 5-2, on liste les cinq paramètres dont quelques-uns seront combinés pour former des paramètres invariants.

Paramètres obtenus par calcul des moments
Excentricité
Longueur axe mineur
Longueur axe majeur
Centroide x
Centroide y

Tableau 5-2 Paramètres obtenus par calcul des moments sur une image.

Pour obtenir les paramètres du tableau 5-2, nous devons disposer d'images distinctes pour l'ensemble des planches images générées par le FLOWCAM. Pour une manipulation donnée, le FLOWCAM produit une série de planches images dont les paramètres sont inscrits dans des fichiers formatés d'extension fcm. Les planches images doivent être décomposées en images distinctes pour que nous puissions extraire des paramètres qui identifieront chaque cellule passant devant les capteurs. La fonction « **imcrop** » permet de résoudre adéquatement ce type de problème. Après avoir séparé chaque image, on effectue certains traitements « filtrage et segmentation » dont l'objectif est d'obtenir uniquement le contour externe de la cellule contenue dans l'image. Ces traitements sont effectués par les fonctions « **imfilter** » et « **histeq** ».

Une fois les images nettoyées de toute information superflue, on effectue le calcul des paramètres sur les moments d'ordre n<3 par la fonction « **imfeature** » sur chacune des formes. La fonction « **imfeature** » retourne différente valeurs de moments.

Tous ces paramètres ainsi que les quatre paramètres cytométriques provenant du fichier de format fcm sont emmagasinés dans une matrice qui prend le nom de « **allparam.mat** ». Cette matrice est ensuite nettoyée pour enlever les lignes contenant des paramètres nuls. On forme alors une matrice globale qui porte le nom de « **allparam2.mat** ».

5.2.4 Procédures et manipulations

Une étape importante dans la reconnaissance de formes est le choix des paramètres qui constitue en grande partie les caractéristiques de performance de la détection et de la classification. Dans le cas d'objets se présentant sous la forme d'une image, il n'existe pas de méthodologie appropriée dans le choix des paramètres. La littérature stipule que nous devons choisir des paramètres qui sont susceptibles d'être invariants tant au niveau translation, rotation et homothétie.

Les paramètres rattachés aux images proviennent d'une combinaison de moments calculés par la fonction « imfeature » de MATLAB. Cette fonction retourne plusieurs moments mais seuls quelques-uns offrent une possibilité d'invariance sous translation, rotation et homothétie.

Pour choisir les paramètres les plus significatifs, il a fallu s'appuyer sur une technique de décorrélation pour obtenir le plus grand nombre de paramètres orthogonaux pouvant caractériser une cellule spécifique.

Quatre paramètres cytométriques ont été choisis comme étant très significatifs pour bien caractériser une cellule ainsi que les paramètres contruits à partir de l'image associée aux paramétres cytométriques. Pour ce faire, la méthode PCA semblait tout indiquée pour permettre une bonne décorrélation des paramètres; cette méthode laissant de côté ceux qui sont inutiles sans affecter de manière appréciable les caractéristiques d'une cellule spécifique. En effectuant plusieurs combinaisons de paramètres et en appliquant la méthode PCA, nous étions ainsi assurés d'obtenir le plus grand nombre de paramètres orthogonaux. À partir de plusieurs listes de paramètres, nous sommes arrivés à déterminer par l'expérimentation laquelle présentait une décorrélation maximale.

Le tableau 5-3 décrit des combinaisons de paramètres qui sont les plus susceptibles de caractériser une cellule particulière de phytoplancton. À l'aide d'un graphique nous statuons sur l'évolution de la décorrélation selon le nombre de paramètres restants via le taux de variance entre ceux-ci. Le calcul de ces paramètres est dépendant de la qualité des images obtenues par le FLOWCAM.

La fonction « imfeature » de MATLAB est un peu limitative dans les paramètres qu'elle fournit car tous ne sont pas des invariants. Nous utilisons uniquement les moments d'ordre 2 concernant la forme de la cellule pour former les combinaisons qui seront des invariants sous translation, rotation et homothétie.

Le fichier « segm.m » que l'on retrouve dans l'annexe permet le calcul des paramètres liés aux images par la méthode des moments d'ordre 2.

Paramètres de caractérisation
Liste 1
Excentricité/chlorophyle peak
Excentricité/phyco peak
axe mineur - execentricité
chlorophyle peak/chlorophyle TOF
excentricité
axe majeur + excentricité
axe majeur
axe mineur
ch peak
ch TOF
Phyco peak
Phyco TOF

Paramètres de caractérisation
Liste 2
Excentricité/chlorophyle peak
Excentricité/phyco peak
axe mineur/axe majeur
chlorophyle peak/Phyco peak
excentricité
axe majeur + axe mineur
axe majeur
axe mineur
ch peak
ch TOF
Phyco peak
Phyco TOF

Paramètres de caractérisation
Liste 3
Excentricité/chlorophyle peak
Excentricité/phyco peak
axe mineur/axe majeur
chlorophyle peak/Phyco peak
excentricité
axe majeur + axe mineur
axe majeur
axe mineur
chloro peak/chloro TOF
ch TOF
Phyco peak/phyco TOF
Phyco TOF

Paramètres de caractérisation
Liste 4
Excentricité
Excentricité/phyco peak
axe mineur/axe majeur
chlorophyle peak/Phyco peak
Centroïde x/centroide y
axe majeur + axe mineur
(axe min/axe maj)*(ph peak/chlo peak
axe mineur
chloro peak
ch TOF
Phyco peak
Phyco TOF

Paramètres de caractérisation
Liste 5
Excentricité
Excentricité/phyco peak
centroide y
chlorophyle peak/Phyco peak
Centroïde x
axe majeur - axe mineur
axe majeur
axe mineur
chloro peak
ch TOF
Phyco peak
Phyco TOF

Tableau 5-3 Combinaison de paramètres pouvant caractériser un objet

Pour construire les listes, on présentait chacune de celles-ci à un décorrélateur et si elle était considérée comme trop corrélée, un nouveau choix de paramètres s'imposait alors et on recommencait la procédure. Sur la figure 5.4, on peut remarquer une diminution du nombre de paramètres selon le taux de variance accepté dans les données lors du traitement par la méthode PCA. Ce taux de variance indique si les axes principaux sont très proches l'un de l'autre. Plus bas sera le taux de variance et plus élevé sera le nombre de paramètres.

Sur la base de ce taux de variance, nous avons choisi la liste dont le nombre de paramètres conservaient une certaine constance jusqu'au taux de 5%. En calculant la moyenne des paramètres restants en fonction de la variance pour chacune des listes, nous sommes arrivés à la conclusion que la liste 4 représente le meilleur choix des paramètres à utiliser pour caractériser une cellule. Les paramètres ainsi énumérés ont alors servi comme paramètres de référence dans la conception du simulateur pour la stratégie un (1).

Variance > %	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
# liste paramètres	Nombre de paramètres décorrélés									
1	8	7	7	7	7	5	5	4	4	3
2	6	6	6	6	6	6	5	5	4	4
3	8	8	8	6	6	6	5	5	5	3
4	8	8	8	8	7	7	5	5	4	4
5	8	7	7	6	6	6	5	5	4	4

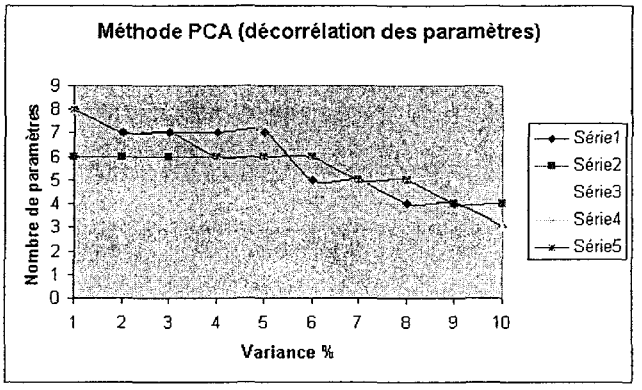


Figure 5.4 Stabilité des paramètres lors d'une décorrélation par méthode PCA

Le tableau 5-4 illustre cette liste de paramètres qui pourrait sembler exhaustive mais une étude plus approfondie pourrait être réalisée afin de vérifier si d'autres paramètres peuvent s'ajouter afin d'obtenir un plus grand nombre de paramètres orthogonaux.

Après décorrélation, nous disposons de huit paramètres complètement orthogonaux entre eux parmi les douze paramètres que nous avons choisis lors de cette expérimentation, en conservant les paramètres dont la variance est plus faible que 5%.

Excentricité
Excentricité/chloro peak
Longueur axe mineur/longueur axe majeur
Chloro peak/phyco peak
Centroide x/ centroide y
Longueur axe majeur+longueur axe mineur
(lng axe majeur/lng axe mineur) *(chloro peak/phyco peak)
Longueur axe mineur
Chloro peak
Chloro TOF
Phyco peak
Phyco TOF

Tableau 5-4 Liste des paramètres images et cytométriques (stratégie 1)

Pour ce qui est de la manipulation de la stratégie deux (2), la liste des paramètres provenant uniquement du fichier fcm contenait un trop grand nombre de paramètres qui étaient corrélés entre eux. Nous avons alors construit une liste combinant des paramètres existant tout en gardant ceux qui étaient disponible. Cela nous donnait 17 paramètres avant décorrélation tout comme dans la stratégie un. Après décorrélation, il restait 12 paramètres qui étaient orthogonaux pour représenter convenablement la cellule d'*Alexandrium tamarense.*

Le tableau 5-5 illustre la liste des paramètres utilisés lors de la simulation de la stratégie deux (2). Cette liste est celle qui nous semble la plus optimisée à partir de nombreux tests de décorrélations.

Cell area
Chlorophyll peak
Chlorophyll TOF
particles per chain
FFT Area
Phyco Peak
Phyco TOF
Feret_max_diameter
Feret_min_diameter
Cellx
Celly
Cell ESD
Cell Area/FFT Area
Chlorophyl peak/Chloro TOF
Phyco peak/ phyco TOF
Feret max/Feret min*chloro peak
(Cell x - Cell y) /Cell ESD

Tableau 5-5 Liste des paramètres cytométrique (stratégie 2).

La détermination de ces deux listes (la liste de paramètres pour la stratégie un (1) et la liste de paramètres pour la stratégie deux (2)) nous amène à l'étape d'apprentissage qui demande de nombreuses manipulations étant donné le plan d'expérience utilisé pour déterminer la meilleure configuration du réseau de neurones. Lors de cette étape d'apprentissage, seule les bases de données #1 et #2 seront nécessaires. Il est d'usage lors de l'apprentissage du réseau de neurones de ne pas apprendre trop exactement tous les exemples mais seulement une partie.

La figure 5.5 est le schéma bloc qui est commun tant pour l'étape d'apprentissage que pour l'étape de classification pour la stratégie un (1).

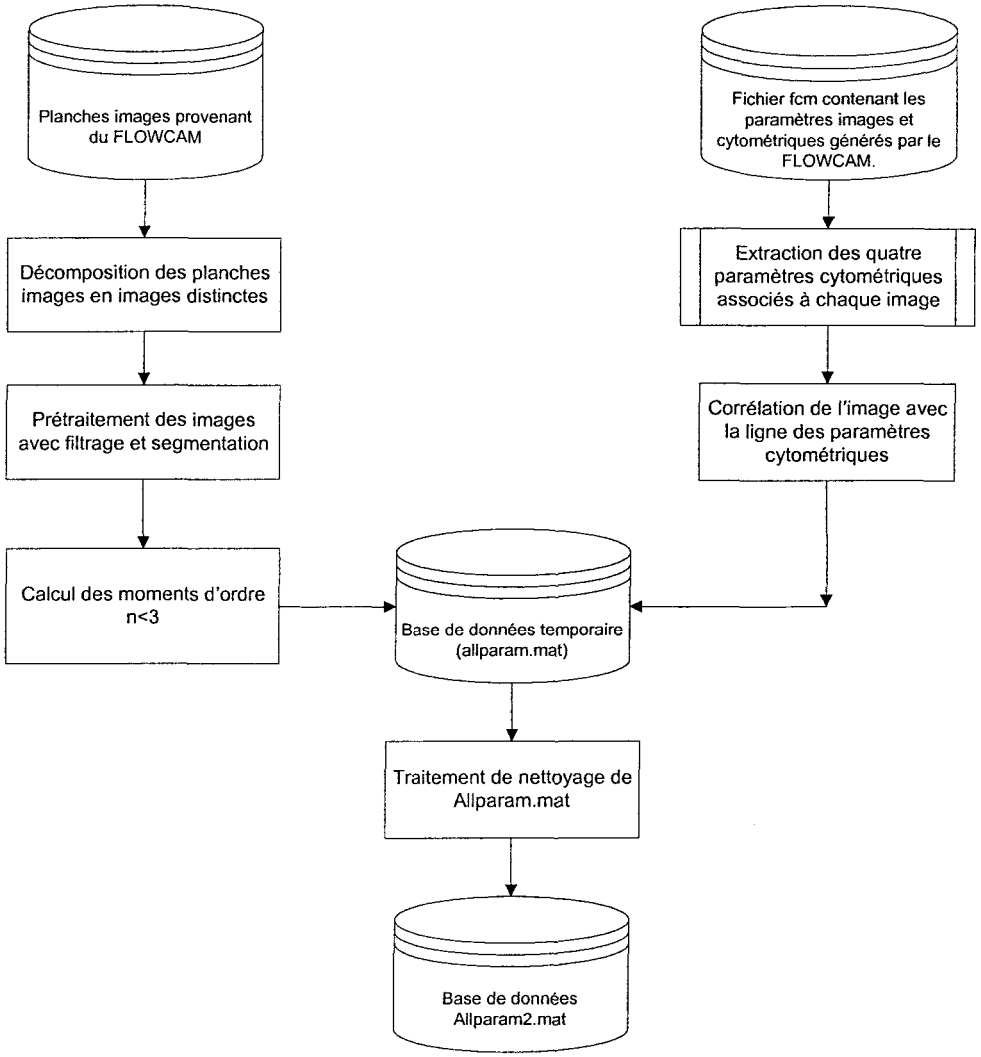


Figure 5.5 Stratégie un (1) - Traitement des images et ajout des paramètres cytométriques

Dans ce diagramme, le nettoyage de la matrice « allparam.mat » permet d'éliminer des images présentant des défauts de structure.

La figure 5.6 est un schéma représentant l'étape d'apprentissage de la stratégie un (1). Une fois l'apprentissage acquis, on présente la base de données #1 et #3 au modèle pour sa validation.

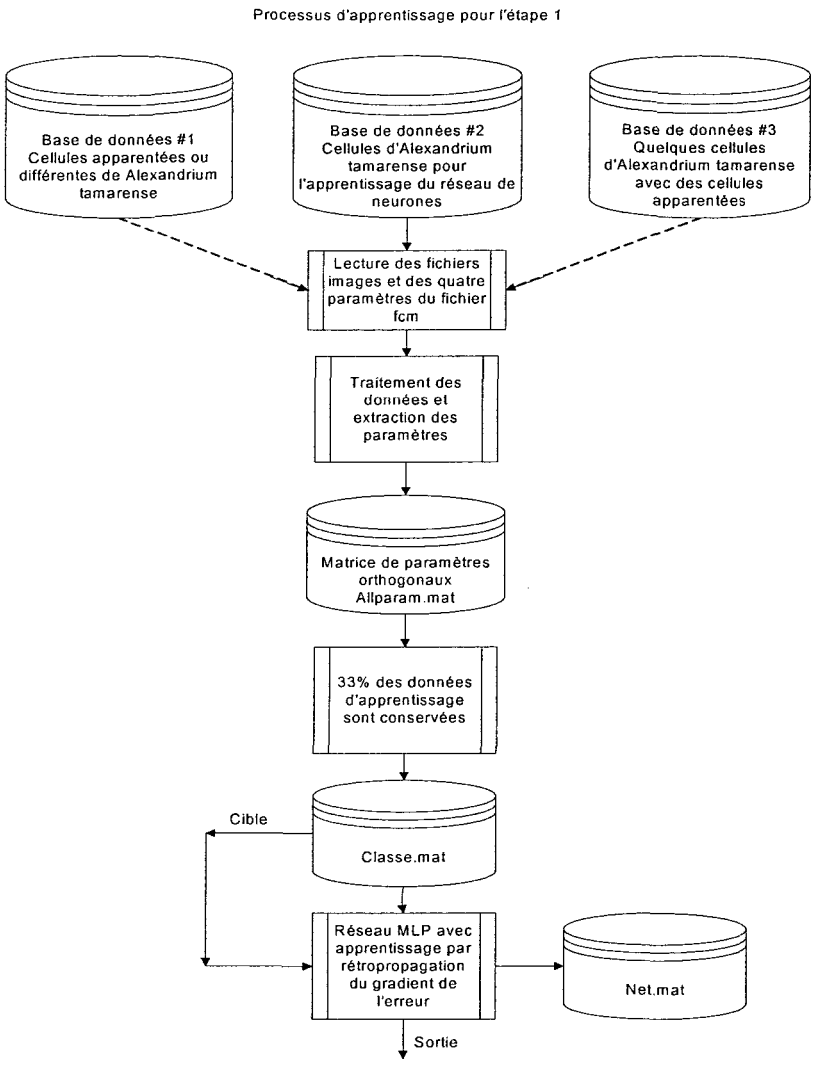


Figure 5.6 Schéma d'apprentissage pour la stratégie 1 (module cyto.m)

Pour la stratégie deux (2), on utilise uniquement les paramètres se trouvant dans le fichier à extension .fcm. Le résultat de la classification est comparé avec la stratégie un (1). C'est le même schéma que la stratégie un (1) sauf que l'étape de traitement des images ne s'y retrouve pas et le nom des modules est différent. La stratégie deux (2) permettra de vérifier s'il est possible d'utiliser uniquement les paramètres du tableau 5-5 pour la caractérisation des cellules à identifier. La figure 5.7 présente une schématisation du module dont le codage se retrouve dans l'annexe sous le nom de « segm_cyto.m ».

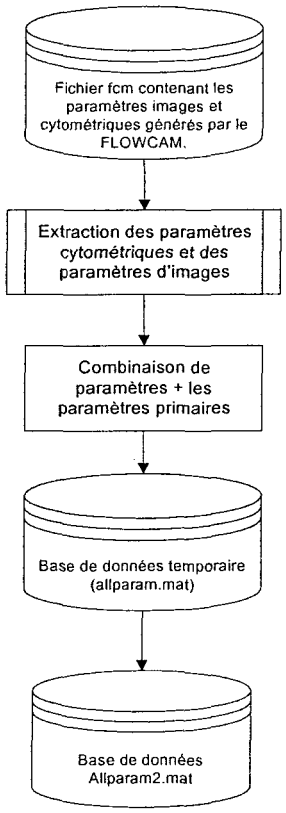


Figure 5.7 Stratégie 2 - Lecture des paramètres du fichier de format fcm

Le fait d'expérimenter avec les données brutes du fichier fcm est le pendant de la stratégie un (1). Cette expérimentation permet de vérifier l'amélioration apportée aux traitements d'images contrairement aux données brutes fournit par le FLOWCAM. Après le prétraitement des données, nous passons à l'étape d'apprentissage du réseau de neurones. La figure 5.8 présente le schéma algorithmique du module qui permet ce type de traitement.

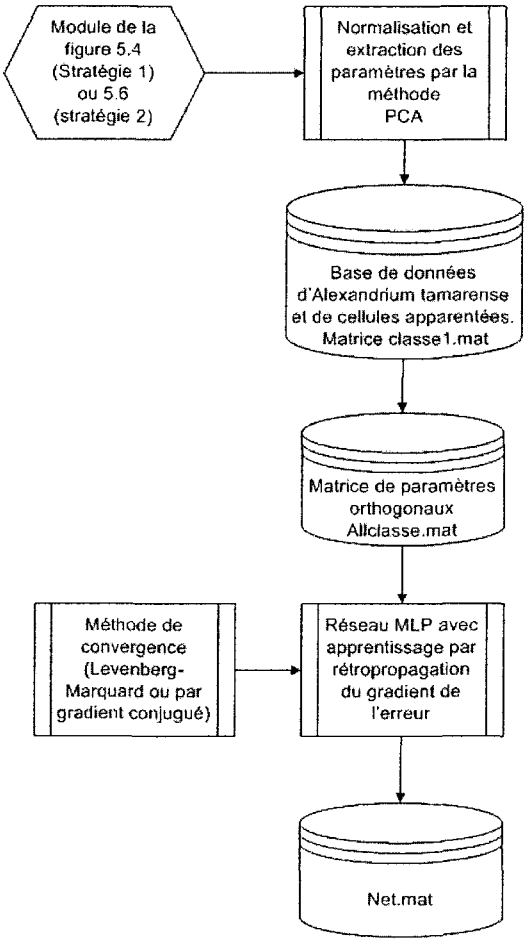


Figure 5.8 Module d'apprentissage du réseau MLP

Pour l'étape d'apprentissage, on tire au hasard 33% des données provenant de la base de données #1 et #2 qui contiennent des cellules d'*Alexandrium tamarense* et des cellules apparentées ainsi que des cellules étrangères. Les données ainsi triées seront annotées de la valeur +1 pour *Alexandrium tamarense* et –1 pour les autres cellules. Cette nouvelle base de données ainsi formée passe dans le module de prétraitement de la figure 5.5 pour ensuite appliquer la matrice « allparam2.mat » dans le module d'apprentissage. Ce module se retrouve sous le nom de « apprentissage.m » de l'annexe. La classe se retrouve dans une base de donnée qui a pour nom « **classe1.mat** ». On effectue par la suite une étape de normalisation des données pour que celles-ci évoluent entre les valeurs –1 et +1 avant de les appliquer à la méthode PCA. Cette procédure a pour objet d'enlever toutes corrélations qui existeraient entres les paramètres. Ces paramètres réduits en nombre seront les seuls qui serviront au classificateur. Cette nouvelle base de données qui regroupera la classe sous étude sera présentée au réseau MLP lors de l'étape d'apprentissage.

Le choix du réseau de type MLP pour cette expérimentation a été justifié par sa fiabilité comme outil de classification. Nous utilisons également le mode d'apprentissage supervisé à cause du peu de données dont nous disposions lors de cette expérimentation. Par contre, si nous voulons faire apprendre au réseau une nouvelle classe, le temps d'apprentissage augmenterait comparativement à l'apprentissage non supervisé qui s'adapte mieux à la présence de nouvelles données pour un temps d'apprentissage moins élevé mais demandant initialement beaucoup plus de données.

La figure 5.9 représente le schéma général du système de reconnaissance d'*Alexandrium tamarense.*

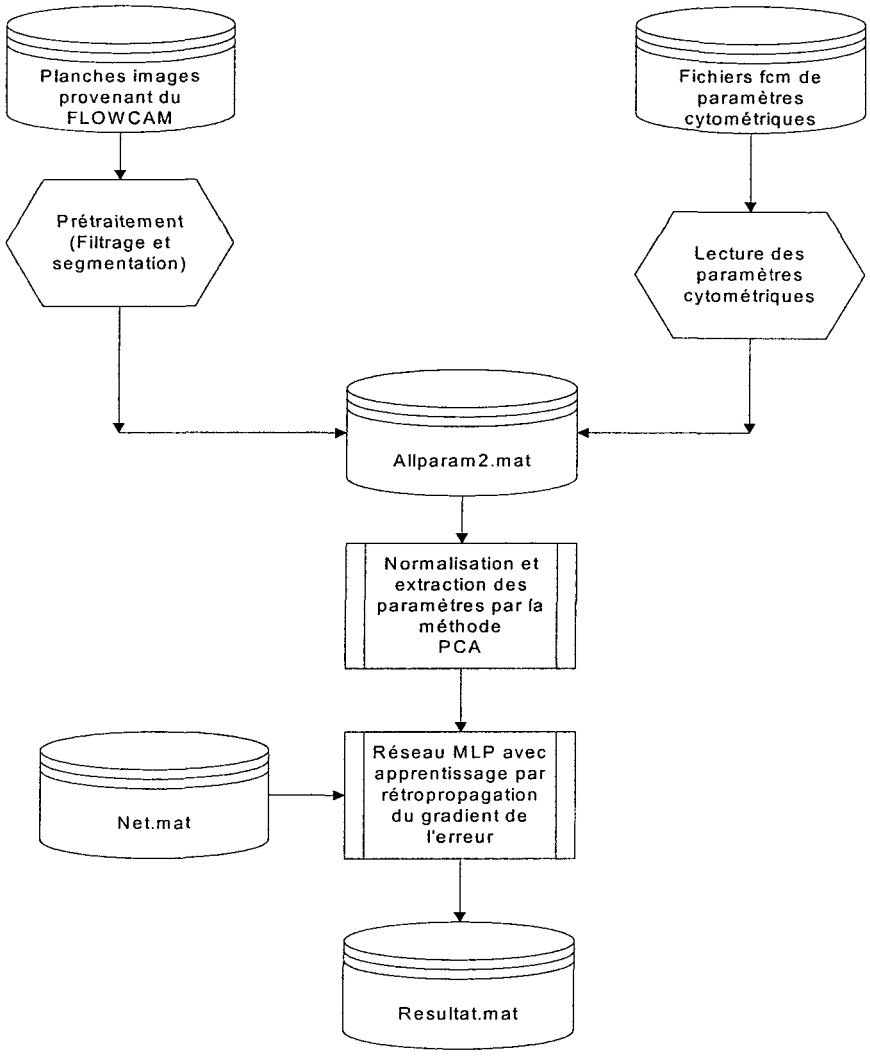


Figure 5.9 Schéma général du système de reconnaissance

On remarquera que la matrice « net.mat » contient toutes les informations concernant la valeur des poids et des biais appartenant à la configuration du réseau de neurones.

Lors de l'étape de classification, nous présentons la base de données #1 et #3 à l'ensemble du système en mode simulation. À la fin de cette expérimentation, nous discutons des résultats obtenus ainsi que des mesures à prendre pour obtenir de meilleures performances en traitement et en précision. Le département de biologie de l'Université Laval nous offre la possibilité de comparer les résultats de la classification selon les analyses par groupements de cellules fournis par le système d'analyse du FLOWCAM.

Pour l'étape d'apprentissage, nous devons déterminer la meilleure configuration à donner pour le réseau de neurones. Le choix des facteurs aura une grande influence sur la configuration du réseau. Comme il n'existe aucune modélisation théorique permettant de configurer un réseau de neurones, nous allons, à partir d'un tableau L9 de Taguchi, construire une série de manipulations (neuf manipulations) qui servira à optimiser la configuration du réseau de neurones selon les résultats de sortie. On vérifiera si la méthode des plans d'expériences est une approche viable selon les résultats produits par la simulation. Nous verrons à la section 5.2.5 quels seront les facteurs à inclure dans la construction du plan d'expériences.

Pour implémenter la matrice d'apprentissage désignée par « net.mat », il existe une fonction qui se nomme « **train** » de MATLAB qui efectue l'entrainement du réseau de neurones.

La configuration de base de ce réseau sera constituée d'une couche d'entrée composée d'autant de neurones qu'il y a de paramètres pour les vecteurs d'entrée, de une ou plusieurs couches cachées et d'une couche de sortie qui sera composée d'un seul neurone dont la valeur de sortie évoluera entre –1 et +1. Dans le cas qui nous préoccupe, nous choisirons un réseau MLP à une ou deux couches cachées de 2N neurones ou N est le nombre de neurones présents à l'entrée. On ne peut pas configurer le réseau MLP au hasard et il est assez problématique de trouver une configuration optimale car il n'y a pas de théorie stipulant le nombre de neurones et de couches cachées nécessaires pour une configuration optimale.

Le résultat de la fonction « TRAIN » est la création d'une matrice qui contiendra la structure entière du réseau MLP. À cette fonction, nous avons expérimenté deux méthodes de convergence, soit la méthode Levenberg-Marquard et la méthode BFGS (algorithme de convergence quasi-Newton) pour le calcul des poids selon la cible présentée en sortie du réseau.

Les valeurs de la matrice « net.mat » sont fixées dans le temps et permettent au réseau de classifier les données qui seront présentées au système lors de la prise de données en temps réel ou provenant d'une base de données en temps différé.

Lors de la simulation de la reconnaissance, les hyperplans seront construits de telle sorte que nous aurons deux classes identifiables, soit la classe +1 pour *Alexandrium tamarense* et finalement la classe zéro pour les cellules n'ayant pas été identifiées lors de la simulation.

Pour comparer les résultats de l'expérience, le logiciel d'analyse associé au FLOWCAM permet de comparer la sensibilité de la simulation ainsi que la précision. Le logiciel présente la particularité d'afficher les données sous forme de nuages de points tel que nous pouvons le voir à la figure 5.10. Cette représentation est la seule qui apparaisse dans l'analyse des données effectuée par le logiciel associé au FLOWCAM.

Nous remarquons trois graphiques : les deux premiers graphiques représentent l'intensité logarithmique de la chlorophyle et l'intensité logarithmique de la phycoérythrine en fonction du diamètre de la cellule tandis que le dernier graphique regroupe les données en fonction de la largeur et de la longueur. Les données sont représentées sous la forme de nuages de points. Les unités de mesure de dimensionnement sont en microns.

La discrimination principale de ce logiciel s'effectue uniquement au niveau de la fluorescence versus le dimensionnememt de la cellule tout en faisant référence à la largeur et à la longueur de la cellule.

Le logiciel associé au FLOWCAM présente certaines particularités interessantes en donnant un minimum d'information sur le type de fluorescence contenu dans une cellule ainsi que le dimensionnement de celles-ci.

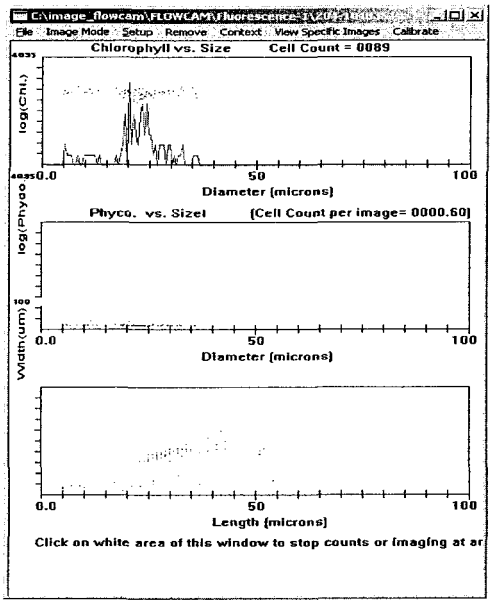


Figure 5.10 Nuage de points de cellules d'Alexandrium tamarense

Pour une culture d'*Alexandrium Tamarence*, nous avons obtenu les résultats apparaissant dans la figure 5.10. La figure 5.11 affiche les résultats provenant de la première base de données ne contenant que des cellules disparates et apparentées à la cellule *Alexandrium Tamarence*. On remarque que la fluorescence due à la phycoérythrine est très faible dans le second graphique de la figure 5.10 comparativement au second graphique de la figure 5.11.

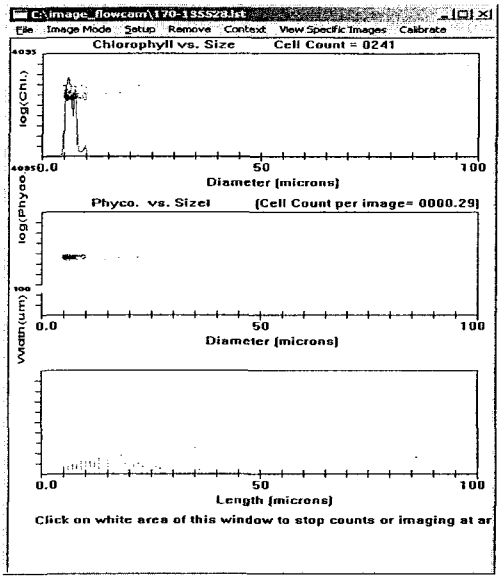


Figure 5.11 Nuage de points de cellules autres qu'Alexandrium tamarense

Dans cette expérience, une série de planches images qui correspondent à une partie de la matrice « allparam.mat » démontre la présence de phycoérythrine plus prononcée qu'avec les planches images d'*Alexandrium tamarense.* Cela laisse supposer à première vue que ce paramètre serait d'une grande importance pour identifier *Alexandrium tamarense* parmi d'autres cellules.

5.2.5 Configuration et optimisation des caractéristiques du réseau MLP

La très faible quantité de données restreint le choix du modèle d'apprentissage à un réseau déterministe à apprentissage supervisé demandant moins d'exemples qu'avec les réseaux à méthodes statistiques. La base de données des exemples, composées de données de la base de données #1 et #2, est présentée plusieurs fois au réseau de neurones suivant un ordre aléatoire selon une loi normale de façon à minimiser au maximum l'erreur quadratique.

Pour obtenir la meilleure configuration du réseau de neurones au niveau de la détection et de la reconnaissance, nous employons la méthode des plans d'expériences de Taguchi pour optimiser le réseau de neurones lors de l'étape d'apprentissage. Le choix du nombre de neurones de la couche cachée ainsi que le nombre de couches cachées nécessaires au problème à l'étude n'est pas déterminable théoriquement. En effet, il n'existe aucune théorie valable spécifiant la configuration optimale qu'il faut donner à un réseau de neurones pour permettre la reconnaissance d'un vecteur de manière rapide et précise. La propriété de généralisation du réseau est également un paramètre à ne pas considérer à la légère. En effet, si le réseau apprend trop parfaitement les schémas directeurs, la généralisation pourrait être médiocre et, de ce fait, il y aurait un mauvais classement d'une nouvelle donnée qui apparaîtrait à l'entrée du réseau.

Une approche originale sur la recherche de la configuration optimale des réseaux de neurones apparaît dans un article concernant les plans d'expériences de Taguchi **[30]**. Après plusieurs manipulations, les auteurs sont arrivés à déterminer les facteurs et les niveaux associés qui seront utilisés dans un plan d'expériences de Taguchi. Les résultats obtenus permettent d'obtenir de manière statistique la configuration optimale selon des tables d'interaction. Il faut choisir les facteurs et le nombre de niveaux d'une manière adéquate pour augmenter la précision de l'étude statistique.

Les facteurs suivants ont été sélectionnés lors de la conception du plan d'expériences: le nombre de neurones de la couche cachée, le nombre de couches cachées, le pas d'apprentissage et le nombre de passes des données présentées au réseau. Des facteurs plus nombreux pourraient être pris en considération pour un travail de recherche plus élaboré.

Pour le réseau MLP, nous avons expérimenté un tableau L9 présenté par le tableau 5-6. Ce tableau sert à produire une série d'expériences qui couvre toutes les possibilités selon les niveaux déterminés dans le tableau L9. De ces expériences, nous pourrons déterminer statistiquement une configuration optimale sur le nombre de couches cachées, le nombre de neurones pour chacune des couches cachées, le pas d'apprentissage et le nombre de passes des données présentées au réseau. Chaque facteur comporte trois niveaux et les paramètres de ces niveaux sont identiques à ceux apparaissant dans l'article des auteurs **[30]**.

Ordre standard	Colonne			
Expérience	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Tableau 5-6 Plan L9 pour optimiser le réseau de neurones

Ce plan indique que pour quatre facteurs constitués chacun de trois niveaux, nous planifions seulement neuf (9) manipulations expérimentales qui détermineront les facteurs causant le plus de variabilités selon les niveaux sélectionnés tout en minimisant les erreurs associées aux bruits. On donne la configuration des niveaux pour chacun des facteurs dans le tableau 5-7. Les échelles pour les niveaux de chacun des facteurs ont été sélectionnées selon l'article concernant la conception d'un plan d'expériences permettant de trouver les niveaux optimaux pour configurer le réseau de neurones **[30]**.

Variables d'expérience et leurs niveaux			
Facteur	Niveau 1	Niveau 2	Niveau 3
A	N/2	N	2N
B	0	N	2N
C	1	100	1000
D	0.1	0.01	0.001

Tableau 5-7 Configuration des facteurs et des niveaux

Le tableau 5-8 founit la description de chacun des facteurs. Seules deux couches cachées sont nécessaires pour le réseau de neurones.

Facteur	Description
A	Nombre de neurones de la couche 1
B	Nombre de neurones de la couche 2
C	Nombre de passes
D	Pas d'apprentissage

Tableau 5-8 Description des facteurs de configuration

Chacun de ces facteurs est annoté dans la table L9 de Taguchi. Les résultats serviront ensuite dans une table de réponse permettant le calcul des effets de chacun des facteurs.

Le tableau 5-9 donne les combinaisons de manipulations qui sont nécessaires pour remplir la table de réponse.

		Facteur				
Numéro d'expérience aléatoire	Ordre standard	A	B	C	D	Résultats
	1	N/2	0	1	0.1	
	2	N/2	N	100	0.01	
	3	N/2	2N	1000	0.001	
	4	N	0	100	0.001	
	5	N	N	1000	0.1	
	6	N	2N	1	0.01	
	7	2N	0	1000	0.01	
	8	2N	N	1	0.001	
	9	2N	2N	100	0.1	

Tableau 5-9 Plan d'expériences L9 d'optimisation du réseau MLP.

Chacune des manipulations est effectuée trois fois et nous calculons la moyenne des résultats pour avoir la certitude qu'aucune erreur ne s'est introduite dans les manipulations. La manipulation expérimentale pour l'étape d'optimisation du réseau de neurones est schématisée par la figure 5.12.

On retrouve trois schémas qui sont annotés processus #1, #2 et #3. Le processus #1 ne se fait qu'une seule fois. L'objectif de ce processus est d'obtenir 33% des données choisies aléatoirement selon une loi normale ; il faut rappeler que les données d'apprentissage proviennent de la base de données #1 et #2. Ce processus s'effectue tant pour la stratégie deux (2) que pour la stratégie un (1). Le processus #2 est celui auquel nous effectuons les modifications de niveau sur la configuration du réseau de neurones.

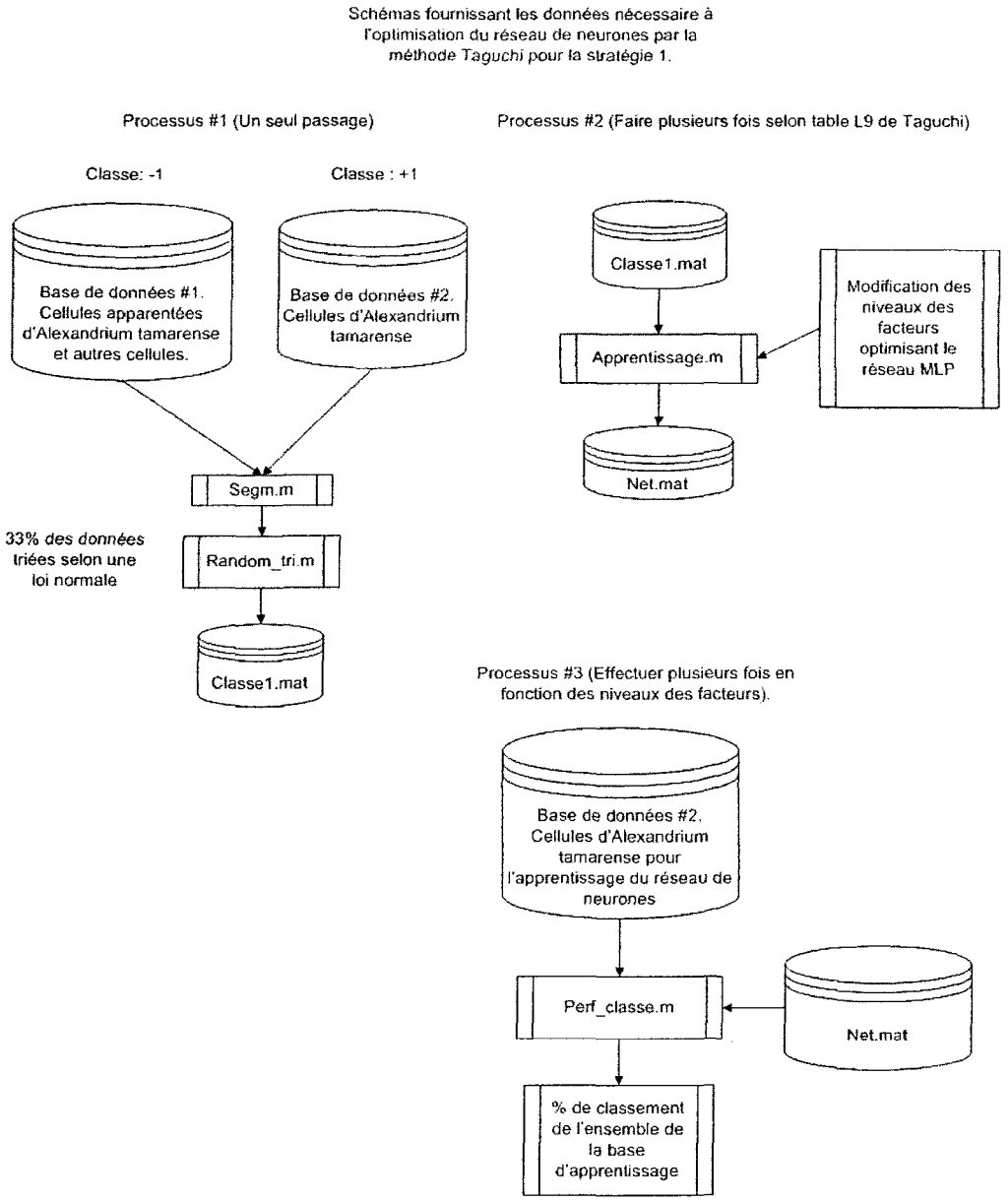


Figure 5.12 Stratégie un (1) : Manipulation pour l'optimisation du réseau de neurones

Le schéma de la figure 5.13 concerne les modules de la stratégie deux (2) qui sont presques identiques au schéma de la figure 5.12 sauf en ce qui concerne le nom des modules.

L'expérimentation sur les deux stratégies s'effectue autant avec la méthode PCA que sans la méthode de décorrélation. Dans l'annexe, le lecteur pourra consulter les sections concernant la codification des modules MATLAB.

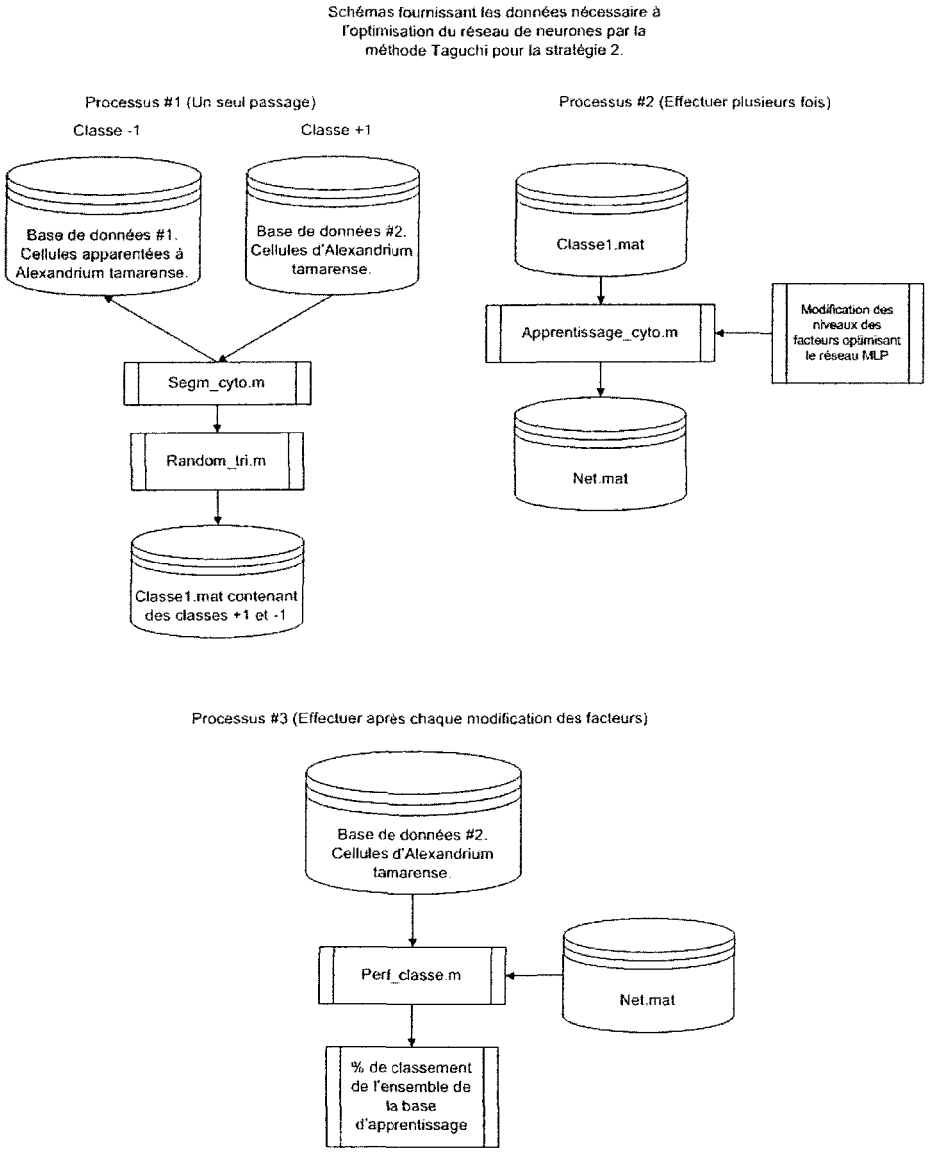


Figure 5.13 Stratégie deux (2) : Manipulation pour l'optimisation du réseau de neurones

Après chaque modification du processus #2, nous lançons le processus #3 pour déterminer le pourcentage de reconnaissance de toutes les cellules de la seconde base de données qui contient les cellules d'*Alexandrium tamarense*.

Le processus d'apprentissage s'effectuera avec l'algorithme de convergence de type quasi-Newton qui se nomme « trainbfg » sous MATLAB. Cet algorithme de convergence est autant utilisé que l'algorithme de Levenberg-Marquardt. La convergence est plus lente mais le résultat précis comparativement à l'algorithme de Levenberg-Marquardt qui approxime une matrice hessienne nécessaire au calcul de la convergence du réseau de neurones.

Dans les deux stratégies, le nombre de paramètres présentés à l'entrée du réseau est identique, soit huit (8) paramètres. Lors de la détermination de la meilleure configuration du réseau de neurones, nous estimons qu'une seule manipulation avec les paramètres de la stratégie deux (2) sera nécessaire pour remplir le plan d'expériences de Taguchi.

En ayant trouvé quels sont les paramètres qui serviront lors de l'expérimentation, nous sommes en mesure d'effectuer une étape d'apprentissage du réseau de neurones MLP sur les données préalablement sélectionnées dans les bases de données #1 et #2 qui contiennent des cellules d'*Alexandrium tamarense* et des cellules apparentées (voir figure 5.14). Sur la totalité des cellules représentées par les deux bases de données, nous avons choisi de manière aléatoire 33% des données que nous avons ensuite appliquées au réseau de neurones avec apprentissage supervisé.

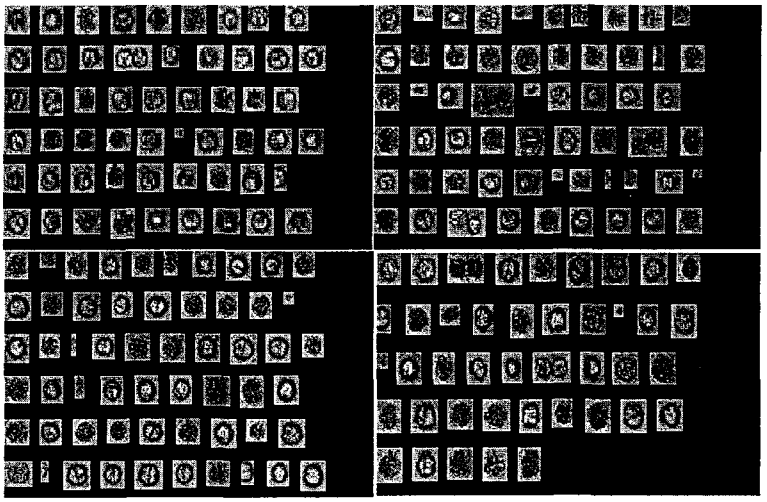


Figure 5.14 Les cellules de la base de données d'apprentissage

Dans le logiciel MATLAB, nous disposons de plusieurs algorithmes de convergence et nous en avons expérimenté deux, dont l'un (Levenberg-Marquardt) convergeait plus rapidement que l'autre (BFGS, algorithme de convergence quasi-Newton). L'algorithme de convergence Levenberg-Marquardt est plus susceptible d'être utilisé pour une application en temps réel et celui-ci est applicable uniquement si la configuration du réseau n'est pas trop complexe. En utilisant au maximum 250 itérations pour l'apprentissage du réseau, nous avons obtenu des résultats qui permettaient une conclusion facile, à savoir que la configuration optimale du réseau donne les résultats apparaissant dans le tableau 5-10.

Nous avons évité la séquence aléatoire car nous avions l'assurance que le temps n'agirait aucunement dans l'obtention des résultats.

err = 0.01	Facteur de configuration du réseau sans décorrélation				
Ordre	Stratégie 2 avec trainbfg				
standard	A	B	C	D	Résultats
					% classé
1	9	0	1	0.1	97.81
2	9	18	100	0.01	94.54
3	9	36	1000	0.001	97.27
4	18	0	100	0.001	96.72
5	18	18	1000	0.1	96.17
6	18	36	1	0.01	97.27
7	36	0	1000	0.01	96.72
8	36	18	1	0.001	100.00
9	36	36	100	0.1	95.56

Tableau 5-10 Table de réponse avec convergence quasi-Newton

En effet, dans le plan d'expériences, il est établit que nous devons mélanger l'ordre des expériences pour relever toute corrélation avec le temps. De ce fait, nous pouvons savoir si le temps joue un rôle sur les facteurs, ce qui n'est pas le cas pour notre expérimentation car nous savons que le traitement de nos données est indifférent au passage du temps.

La table de réponse montre que la manipulation huit (8) donne 100% de bonne classification, laissant sous-entendre que nous n'avons pas besoin d'aller plus loin dans les statistiques des plans d'expériences. La manipulation huit (8) représente une forte possibilité de configuration optimale du réseau MLP. La configuration de ce réseau est alors composée comme suit : 36 neurones pour la première couche cachée, 18 neurones pour la seconde couche cachée et un pas d'apprentissage de 0.001 s'effectuant en une passe.

La stratégie deux (2) a été expérimentée avec 17 paramètres qui ont été présentés au réseau sans effectuer de décorrélation par la méthode PCA. Nous avons également expérimenté avec l'algorithme de convergence Levenberg-Marquardt et nous avons obtenu les résultats suivants apparaissant dans le tableau 5-11.

err = 0.01	Facteur de configuration du réseau sans décorrélation				
Ordre	Stratégie 2 avec Levenberg-Marquardt				
standard	A	B	C	D	Résultats
					% classé
1	9	0	1	0.1	1.09
2	9	18	100	0.01	8.74
3	9	36	1000	0.001	98.91
4	18	0	100	0.001	1.64
5	18	18	1000	0.1	75.96
6	18	36	1	0.01	98.91
7	36	0	1000	0.01	
8	36	18	1	0.001	
9	36	36	100	0.1	

Tableau 5-11 Table de réponse avec Levenberg-Marquardt

Cette méthode de convergence est rapide quoique demandant énormément de ressources mémoire pour l'ordinateur. Nous avons été dans l'impossibilité pour le moment d'essayer les configurations sept (7), huit (8) et neuf (9) du tableau 5-11 étant donné que l'ordinateur indiquait un arrêt du programme pour manque de ressources mémoire.

Les résultats ne sont pas assez probants pour que nous utilisions cette méthode de convergence, surtout pas avec la dimension de la base d'apprentissage. Nous avons donc conservé l'algorithme de convergence quasi-Newton pour la totalité des manipulations dans cette recherche. Aussitôt que le réseau devient moindrement complexe, cet algorithme de convergence devient moins performant en vitesse d'apprentissage et trop gourmand en ressources mémoire.

La convergence était rapidement atteinte pour les manipulations un (1) et quatre (4) mais présentait un taux de classification très médiocre.

Après avoir appliqué la configuration déterminée par la manipulation huit (8) du tableau, nous sommes passés à l'étape de classification sur la base de données #1 et #3 tant pour la stratégie un (1) que pour la stratégie deux (2).

5.3 Conclusion

Toutes les étapes de la simulation autant pour la stratégie un (1) que pour la stratégie deux (2) ont été réalisées avec succès sauf pour l'expérimentation concernant les plans d'expériences avec la méthode de convergence Levenberg-Marquardt. Dans cette étape, nous avons eu quelques problèmes concernant les ressources mémoire de l'ordinateur. Ce problème nous a empêché d'obtenir les résultats des lignes 7, 8 et 9 de la table de résultat du tableau 5-11. La méthodologie a été suivie à la lettre et semble adéquate pour offrir une simulation efficace pour la détection et la reconnaissance d'une cellule spécifique contenant une toxine. Dans la section analyse des résultats, nous discuterons plus à fond des résultats obtenus pour la stratégie un (1) et la stratégie deux (2).

CHAPITRE 6

ANALYSE DES RÉSULTATS

6.1 Préambule

Les résultats concernant la détection d'une cellule d'*Alexandrium tamarense* parmi d'autres cellules distinctes ou apparentées ne permettent pas de conclure avec certitude que la méthode des moments est adéquate en fusion avec les paramètres cytométriques choisis pour la stratégie un (1). Des deux stratégies, la stratégie deux (2) fut la plus efficace et pourtant elle utilise uniquement des paramètres provenant du FLOWCAM sans passer par une méthode de décorrélation.

Pour la configuration du réseau de neurones, la méthode des plans d'expériences à donnée de bons résultats mais il ne faudrait pas statuer sur une réussite complète de cette méthode dans la recherche d'une configuration optimale. Il faudrait effectuer de nombreuses expériences pour confirmer cette méthode d'optimisation. C'est une approche intéressante qui s'ajoute à la panoplie théorique sur la recherche d'une meilleure configuration des réseaux de neurones.

Pour la décorrélation des paramètres, la méthode PCA utilisée pour les deux stratégies à donnée sa pleine efficacité même si celle-ci n'était pas nécessaire dans la stratégie deux (2). C'est une méthode très fonctionnelle mais il se peut que des informations importantes soient perdues durant le traitement de décorrélation.

Pour cette étape finale de notre recherche, la conclusion qui s'impose est que la qualité de la détection et de la reconnaissance d'algues toxiques par fusion de données entre des paramètres cytométriques et des paramètres invariants obtenus par traitement d'image par calcul des moments est discutable tant par le choix des paramètres, le choix du modèle ainsi que la configuration du réseau de neurones utilisé. Ce projet de recherche demeure néanmoins intéressant dans son approche par l'aspect fusion de données qui représente certainement une voie de recherche à suivre.

6.2 Le choix des paramètres

Au cours de l'expérimentation, nous avons vu que le choix des paramètres pouvait représenter une étape qui était des plus discutable car rien ne laissait présager quels étaient les paramètres qui pouvaient déterminer en grande partie les caractéristiques de performance du simulateur. Lors de cette étape, nous recherchions de nombreux paramètres pour ensuite les décorréler pour obtenir la meilleure configuration possible. La méthodologie consistant à combiner de paramètres n'était peut-être pas adéquate. Il est possible que nous ayons manqué ce but et que cela fausse la conclusion sur l'efficacité de la méthode des moments d'ordre n avec $n<3$ lors de l'étape d'extraction des paramètres sur les images du FLOWCAM. La fusion des quatre paramètres cytométriques avec les paramètres obtenus par la méthode des moments ne semblait pas aussi efficace qu'elle le laissait croire. La stratégie deux (2) a été, par contre, plus efficace que la stratégie un (1) en ce qui concerne la reconnaissance d'une cellule distincte.

Les raisons à cela sont attribuables probablement aux paramètres images générés par le FLOWCAM dont on retrouvait un paramètre calculé par FFT. La liste de paramètres générée par le FLOWCAM, dont la plupart présentaient des caractéristiques invariantes, semblait donner d'assez bons résultats et pourrait expliquer le taux d'erreur nul que nous avons obtenu lors de la détection effectuée à l'aide de la stratégie deux (2). Le choix du type de réseau de neurones n'est peut-être pas étrangé à ce fait. La méthode de décorrélation semblait également inutile lors de l'expérimentation de la stratégie deux (2) qui donnait un piètre résultat lorsque nous utilisions la méthode PCA (voir tableau 6-1).

6.3 Configuration du reseau MLP.

Pour obtenir une bonne discrimination, il faut que le réseau de neurones soit bien configuré. Si celui-ci apprend trop parfaitement, les hyperplans sont trop rapprochés et une perte de généralisation en découle. La méthode des plans d'expériences de Taguchi semble donner d'excellents résultats concernant la recherche d'une configuration optimale du réseau de neurones selon le minimum de bruit associé au choix des facteurs et des niveaux. Il se peut que les résultats concernant la configuration du réseau de neurones ne soient pas optimaux mais à la vue des résultats obtenus et avec la vitesse à laquelle la détection s'est effectuée, nous pouvons conclure, s'en trop nous tromper, à une bonne optimisation du réseau.

6.4 Résultats de la simulation

Sur les deux stratégies qui ont été expérimentées, des résultats intéressants ont été obtenus (voir tableau 6-1) mais demanderaient à être validés de manière plus approfondie, par exemple en utilisant d'autres banques de données pour remplacer les base de données #1 et #3. Comme nous ne disposions pas d'un ensemble significatif de données, nous sommes restés dans l'incertitude quant à statuer définitivement sur la précision de la classification avec la stratégie #1. Les résultats sont plus encourageants pour la stratégie #2 mais il ne faudrait pas sauter trop rapidement aux conclusions car la quantité de données que nous avions lors de cette expérimentation ne permet pas une conclusion définitive.

Nombre d'image de cellule: 6548				
Paramètres	Stratégies	Nb Cell. Alex	Nb de cell	Total
		détecté	fausse alarme	
Avec décorrélation	Stratégie 1	1	2	3
Avec décorrélation	Stratégie 2	0	121	121
Sans décorrélation	Stratégie 1	0	16	16
Sans décorrélation	Stratégie 2	1	0	1
Nb d'Alexandrium dans les bases de données #1 et #3 : 1/6548				

Tableau 6-1 Résultats de la simulation pour les deux stratégies

Les résultats que nous avons obtenus avec la stratégie deux (2) semblent indiquer que pour un nombre très restreint de paramètres, la méthode PCA semble inutile. Les résultats étaient meilleurs en utilisant les paramètres provenant du fichier généré par le FLOWCAM sans utilisation de la méthode PCA.

Pour le cas de la stratégie #1 dont les fondements s'appuyaient sur la détermination de paramètres par un traitement des images avec fusion de quatre paramètres cytométriques, nous avions obtenu deux fausses alarmes et une cellule toxique détectée. Pour la stratégie deux, nous avons obtenu une cellule toxique détectée et aucune fausse alarme comparativement aux paramètres générés par le calcul des moments sur les images. La stratégie 2, sans utilisation de la méthode PCA, permettait de reconnaître et d'identifier une cellule d'*Alexandrium tamarense* parmi un ensemble de 6548 images combinant la base de donnée #1 et #3. Les raisons à cela proviennent certainement des paramètres enregistrés par les capteurs du FLOWCAM, on retrouve des paramètres calculés sur chacune des images, par exemple la surface de la cellule, la transformée de Fourier de l'image ainsi que des paramètres de dimensionnement de la cellule. Pour la stratégie deux (2), les paramètres n'ont pas été décorrélés et cela peut expliquer le meilleur rendement de la stratégie deux (2) sur la stratégie un (1) parce que le nombre de paramètres était assez réduit et que lors du passage par le décorrélateur, nous perdions des informations importantes qui caractérisaient la cellule.

Les résultats de la stratégie #1 ne sont donc pas très concluants car, en plus de détecter une cellule toxique, il y avait également détection de deux cellules apparentées qui sont reconnues comme transportant des toxines, ce qui est totalement faux.

6.5 Résultats obtenus par le logiciel associé au FLOWCAM

Le FLOWCAM ne dispose que d'un système de reconnaissance de base limité à des paramètres de dimension et de fluorescence. Nous avons été en mesure de comparer la qualité de la détection et de la classification entre notre système de détection et celui du FLOWCAM et nous voyons sur la figure 6.1 que l'information se perd. En effet, la cellule deux (2) correspondant à *Alexandrium tamarense* est perdue dans le nuage de points représentant diverses caractéristiques qui se confondent.

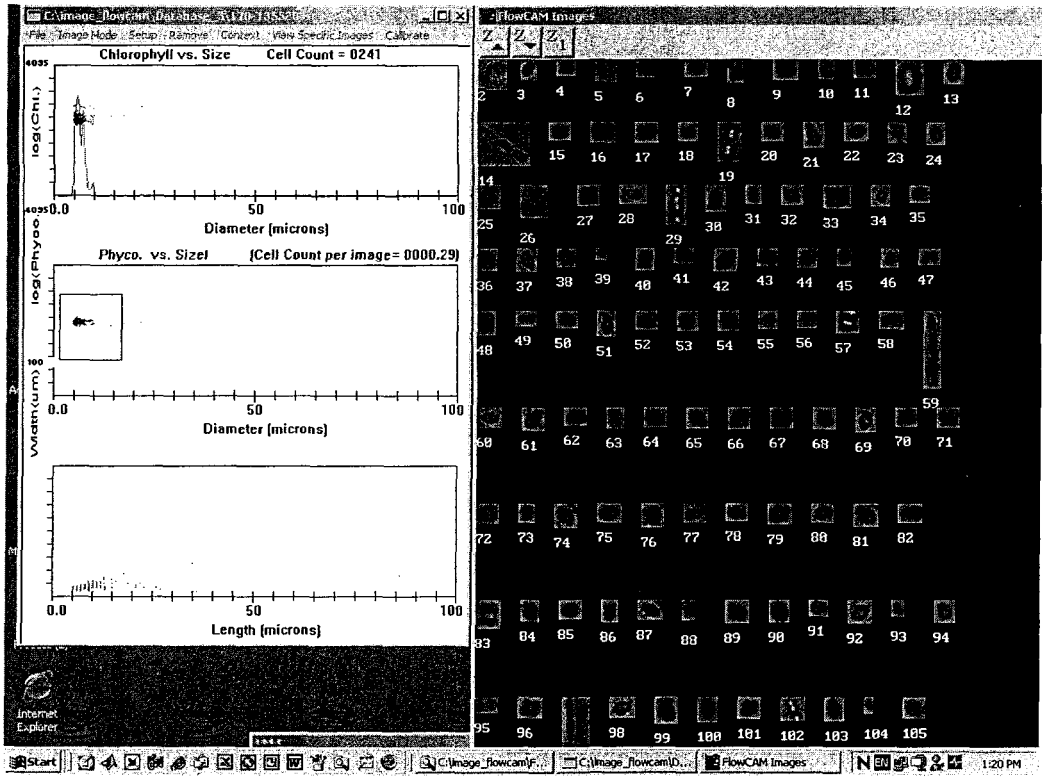


Figure 6.1 Nuage de cellules indistinctes

Normalement, le niveau très bas de phycoérythrine aurait permis une certaine différentiation mais ce n'était pas le cas pour cette planche image. À moins d'une erreur et loin de mettre en doute les capacités de reconnaissance du logiciel d'analyse du FLOWCAM, le modèle que nous avons conçu en fonction des paramètres générés par le FLOWCAM permettrait une bonne détection parmi un ensemble de cellules apparentées ou non. Pour être en mesure de vérifier la validité du système de reconnaissance, une expérimentation plus approfondie serait nécessaire, par exemple avec des moments d'ordre plus élevés et des données en plus grands nombres. Nous laissons ce soin pour des recherches ultérieures.

6.6 Conclusion

La situation, en ce qui a trait aux systèmes existants pour la reconnaissance automatisée du phytoplancton, est en plein développement. Il y a néanmoins des difficultés inhérentes quant à la précision de la classification et de la détection à partir du calcul d'invariants par la méthode des moments d'ordre $n<3$.

Les objectifs de ce projet de recherche ont été concluants en ce qui a trait à la reconnaissance automatisée d'une espèce spécifique de phytoplancton en utilisant uniquement les paramètres cytométriques contrairement à la fusion de données de la stratégie #1.

Dans cette recherche, nous avons expérimenté deux stratégies, la première étant une fusion de données obtenues par traitement d'images associées à des paramètres cytométriques générés par le FLOWCAM tandis que la seconde était l'utilisation de la totalité des paramètres cytométriques sans procéder au calcul des moments d'ordre n<3 sur chacune des images. Pour les deux approches, la discrimination des classes a été effectuée par un réseau de neurones. L'optimisation du réseau de neurones (nombre de couches cachées, nombre de neurones pour chacune des couches cachées) a été obtenue par une méthode statistique : la méthode des plans d'expériences de Taguchi.

Dans les deux stratégies, nous avons émis l'hypothèse que les données présentaient une distribution normale. À partir de cette distribution, 33% des données provenant de la base de données #1 et #2 ont été choisies de manière aléatoire pour être présenté ensuite au réseau de neurones lors de l'étape d'apprentissage. Nous avons ensuite présenté les données provenant de la base de données #1 et #3 au système de reconnaissance pour les deux stratégies.

Les résultats ainsi obtenus ont permis de conclure, à partir des échantillons que nous avions à notre disposition, que le calcul des paramètres par la méthode des moments d'ordre n<3 dans la stratégie #1 était inadéquate par rapport à la stratégie #2. Après une recherche infructueuse, aucun système de reconnaissance du phytoplancton associé avec le FLOWCAM n'est en opération et il fut alors difficile de pouvoir comparer d'une façon appropriée les résultats obtenus dans cette recherche.

Il appert à la vue de l'expérimentation que la conception d'un système de détection d'algues toxiques est une solution envisageable en utilisant uniquement les paramètres générés par le FLOWCAM pour la détection en différé. La recherche d'invariants par la méthode des moments d'ordre n<3 n'a pas démontré sa puissance dans la stratégie #1, peut-être parce que nous avons utilisé une valeur de n trop petite pour le calcul des moments d'ordre n. C'est une voie de recherche à prendre en considération pour un travail futur concernant la recherche d'invariants de forme.

Il est plus que probable que si nous avions disposé de la forme des signaux pour chacune des cellules détectées par le cytomètre, nous aurions obtenu un résultat plus probant que celui que nous avons eu pour la stratégie #1 lors de l'expérimentation. Il serait alors possible d'utiliser la reconnaissance de signaux qui proviendraient du FLOWCAM. Aujourd'hui, les méthodes de détection s'attaquent à la possibilité d'attacher un virus à une cellule spécifique et ensuite détecter celle-ci par un procédé faisant appel à la reconnaissance de l'ADN. On cherche également un moyen d'éliminer les « blooms » à l'aide d'agents chimiques ou biologiques qui s'attaqueraient uniquement aux algues toxiques.

La recherche en reconnaissance de formes est encore et sera, pour une longue période, un domaine à creuser car, depuis 50 ans, nous n'avons pas encore résolu complètement le problème d'invariance dans les données présentées à un classifieur. La technique de classification par réseaux de neurones demeure encore une voie de recherche en plein développement et il n'existe aucune théorie permettant d'évaluer la configuration optimale d'un réseau indépendamment de l'application envisagée.

La réalisation de la partie expérimentale de cette recherche a été rendue possible grâce aux données du FLOWCAM que le département de biologie de l'Université Laval a mis à notre disposition. N'eut été de M. Maurice Levasseur et de ses coéquipiers de recherche, nous n'aurions pas été en mesure d'amener à bon port ce projet de recherche.

6.7 Suggestions de travaux futurs

Pour la détection d'algues toxique avec des données provenant du FLOWCAM, il est assuré que ce travail présente une voie de recherche dans la reconnaissance et la détection en différé au moyen des paramètres générés par le FLOWCAM. Il faudrait avoir la possibilité d'expérimenter avec d'autres sources de données pour conclure sur la démarche expérimentale.

Une étude plus approfondie serait également nécessaire pour l'extraction des paramètres provenant du traitement même de l'image de la cellule par les moments d'ordre n avec n plus grand que trois (3). On peut toujours expérimenter avec la transformée de Fourier de l'image pour obtenir d'autres invariants.

Une autre voie de recherche serait d'expérimenter la stratégie un (1) et deux (2) avec un réseau auto-associatif tel que le réseau de Kohonen sauf ce qui concerne les invariants qui seraient déterminés par la méthode des moments ou par la FFT de l'image.

Finalement, il serait intéressant d'obtenir les signaux cytométriques pour chacune des cellules et utilisé un réseau de neurones qui apprendrait à reconnaître la forme des signaux. Il pourrait découler, de cette expérience, des résultats intéressants sur la détection de plusieurs espèces de manière distinctes.

BIBLIOGRAPHIE

[1] Blasco D., M.Levasseur, R.Gélinas, R.Larocque, A.D. Cembella, B Huppertz, E.Bonneau, 1998. **Monitorage du phytoplancton toxique et des toxines de type IPM dans les mollusques du Saint-Laurent: 1989 à 1994**. Rapport statistique canadien sur l'hydrographie et les sciences océaniques. No.151, 117 p.

[2] *Philippe LEBARON, Laboratoire d'océanographie biologique de Banyuls, CNRS-Université Paris 6, CNRS INFO. Revue de presse.* **Détection rapide et sensible de microorganismes pathogènes dans l'environnement aquatique**. No 388, décembre 2000.

[3] M. Egmont-Petersen, D. de Ridder, H. Handels. **Image processing with neural networks – a review.** Elsevier, The journal of the Pattern Recognition Society, 35 (2002) 2279-2301.

[4] Abdel Belaïd, Yolande Belaïd, **Reconnaissance des formes, méthodes et applications**. InterEditions 1992, Paris.

[5] J. Vives-Rego, P. Lebaron, G. Nebe-Von Caron. **Current and future applications of flow cytometry in aquatic microbiology**. Elsevier, Fems Microbiology Reviews 24 (2000) 429-448.

[6] Robert J. Olson, Alexi Shalapyonok, Heidi M. Sosik. **An automated submersible flow cytometer for analyzing pico- and nanophytoplankton,** FlowCytobot. Elsevier, Deep-sea research (2003).

[7] PF Culverhouse, RG Simpson, R Ellis, JA Lindley, R Williams, T Parisini, B. Reguera, I. Bravo, R. Zoppoli, G. Earnshaw, H. McCall and G. Smith. **Automatic classification of field collected dinoflagellates by artificial neural network.** Mar. Ecol. Prog. Ser. 139 (1-3), 1996 PP.281-287.

[8] Thierry Denoeux . **Pattern classification.** Handbook of neural computation @1997 IOP Publishing Ltd and Oxford University Press, release 97/1.

[9] Robert W. Smith, James F. McNamara and David S. Bradburn. **Pattern classification Techniques Applied to Character Segmentation. Classifiers for Character Segmentation.** Berkeley Recognition Systems.

[10] Farzin Mokhtarian and Farahnaz Mohanna. **Fast active contour convergence through curvature scale space filtering**. University of Surrey,UK, 2002.

[11] *Julian D. Olden* **An artificial neural network approach for studying phytoplankton succession**. Hydrobiologia 436: 131-143, 2000.

[12] Friedrich Recknagel, **Artificial Neural Network model for predicting species abundance and succession of blue-green algae.** Hydrobiologia 349: 47-57, 1997.

[13] D. Dollfus, L. Beaufort **Fat neural network for recognition of position-normalised objects**, Neural Network 12 (1999) 553-560.

[14] G. Dreyfus. **Les réseaux de neurones**. Mécanique industrielle et matériaux, n°51 (septembre 1998).

[15] Steven W. Smith, Ph.D. California Technical Publishing ISBN 0-9660176-3-3 **The Scientist and Engineer's Guide to Digital Signal Processing**. Chapter 26 – Neural Networks (1997).

[16] James L. Mueller and Roswell W. Austin. Volume V, **Ocean Optics Protocols for SeaWifs Validation.** NASA Technical Memorandum, SeaWiFS Technical Report Series Stanford B. Hooker, Editor (March 1992).

[17] Sing-Tze Bow. **Pattern Recognition and Image Preprocessing**. Electrical engineering and electronics, New York : M. Dekker c1992.

[18] Jeanny Hérault, Christian Jutten. **Réseaux neuronaux et traitement du signal**. Traité des nouvelles technologies, série Traitement du signal, Hermes, Paris, 1994.

[19] Adrian Low. **Introductory Computer Vision and Image Processing**. McGRAW-HILL BOOK COMPANY, 1991.

[21] Teuvo Kohonen. (1990), **The self-organising map.** Proceeding of the IEEE, vol. 78, No.9, Septembre 1990, 1464–1480.

[22] Hugues Benoit-cattin, Tarik Zouagui, Christophe Odet, **Une vision fonctionnelle de la segmentation d'images** CREATIS, Unité de recherche CNRS (UMR 5515), affiliée à l'INSERM.

[23] Xiaoou Tang, W. Kenneth Stewart, Luc Vincent, He Huang, Marty Marra, Scott M. Gallager and Cabell S. Davis. **Automatic plankton image recognition**, Artificial Intelligence Review 12: pp. 177-199, 1998.

[25] Lee DS, Srihari SN, Gaborski R. **Bayesian and neural network pattern recognition: a theoretical connection and empirical results with handwritten characters.** In : Sethi IK and Jain AK (eds), North-Holland, 1991

[27] J. Feldman, M. A. Fanty, and N. H. Goddard. **Computing with structured neural networks.** Computer, 21(3):-103, March 1988.

[28] Anil K. Jain, Fellow, IEEE, Robert P.W. Duin, and Jianchang Mao, Senior Member, IEEE. **Statistical Pattern Recognition: A Review.** IEEE transactions on pattern analysis and machine intelligence, VOL.22, NO.1, January 2000.

[29] Olson RJ, Chisholm SW, Frankel SL, and Shapiro HM (1983). **An inexpensive flow cytometer for the analysis of fluorescence signals in phytoplankton: Chlorophyll and DNA distributions**. J. EXP. MARINE BIOLOGY ECOL, 68, 129-144.

[30] Young-Sang Kim, Bong-Jin Yum, 2003. **Robust design of multilayer feedforward neural network: an experimental approach.** Engineering Applications of Artificial intelligence, Elsevier.

[31] Taguchi, G., 1987 **Systems of Experimental Design,** Vol. 1 & 2. UNIPUB/Kraus International Publications, New York.

[32] Laforge, Hubert, 1981 **Analyse multivariée pour les sciences sociales et biologiques avec applications des logiciels BMD, BMDP, SPSS, SAS,** Édition Étude Vivantes, Montréal.

[33] R. Duda and P. Hart. **Pattern recognition and Scene Analysis.** John Wiley and sons, New York, 1973.

[34] E. Davalo, P. Naïm. **Des réseaux de neurones**. Éditions Eyrolles 1998.

[35] Christopher M. Bishop. **Neural Networks for Pattern Recognition.** Oxford University Press, 1995.

[36] R. R. Coifman and N. Saito, **Constructions of local orthonormal bases for classification and regression**, Comptes Rendus Acad. Sci. Paris, Serie I, vol. 319, no. 2, pp. 191-196, 1994.

[37] R. E. Learned and A. S. Willsky. **A wavelet packet approach to transient signal classification**, Appl. Comput. Harmonic Anal., vol. 2, no. 3, pp. 265-278, 1995.

[38] Michel Misiti, Yves Misiti, Georges Oppenheim, Jean-Michel Poggi **Wavelet Toolbox for use with MATLAB**, July 2002 Online only Revised (Release 13) Version 2.2 doc@mathworks.com

[39] P.G.J. Lisboa **Neural Network current applications**. 1992, Chapman & Hall.

[40] McCulloch, W.S. and W. Pitts (1943), **A logical calculus of the ideas immanent in nervous activity.** Bulletin of Mathematical Biophysics 5, 115-133. Réimpression chez Anderson and Rosenfeld (1988).

[41] Minsky, M.L. and S.A. Papert (1969) **Perceptrons.** Cambridge, MA: MIT Press. Expanded Edition 1990.

[42] Arkadev A. G. , Braverman E. M. **Teaching Computers to Recognize Patterns** Academic Press, London and New York – 1967.

[43] M. F. Wilkins, Lynne Boddy, C. W. Morris, and R. R. Jonker. **Identification of Phytoplankton from Flow Cytometry Data by Using Radial Basis Function Neural Networks.** Applied and Environmental Microbiology, Oct. 1999, p. 4404-4410.

[44] Howard Demuth, Mark Beale **Neural Network Toolbox for use with MATLAB**, January 1998 Online only Revised (Release 13) Version 2.2 doc@mathworks.com

ANNEXE A

Modules communs aux deux stratégies

MODULE SEGM.M

```
--------------------------------------------------------------------------------
% ------------------------------------------------------
% ------------- Module SEGM.M ------------------
% ------------------------------------------------------
%
% Richard Lepage (Maitrise en ingénierie)
% Université du Québec à Rimouski
% Ce logiciel est concu selon des fonctions de MATLAB
%
% Librairie: network, image processing, statistic
%
%
% Création des paramètres nécessaires à la reconnaissance de l'image selon
% les mesures de moments effectuées sur le contour de l'image
%
%% Prétraitement (Filtrage, segmentation de l'image et extraction des parametres).
%  Ce module est utilisé tant pour l'étape d'apprentissage que pour l'étape de classification
%
% ------------------------------------------------------------------------
% --------- Pour chacune des planches d'extension .tif,
% ----------séparation individuelle des images de la planche.
% ------------------------------------------------------------------------
%

%On efface le workspace et l'écran des résultats
clear;clc;
% Les planches d'images d'une expérience sont en format .tif
pathimage='c:/image_flowcam/*.tif';
% Fichier format texte des parametres cytométriques
pathcytometre='c:/image_flowcam/*.fcm';
% On efface tout fichiers avec extension .jpg
delete *.jpg;
% On recherche le nom de toutes les planches images .tif
D=dir(pathimage);
% datacyto contient la liste de tous les fichiers .fcm
```

```
datacyto=dir(pathcytometre);
% Chemin des fichiers images .tif et .fcm
pathfilename='c:/image_flowcam/';
extension='.jpg';
%Indexation des images individuelles
nbcell=1;
noexp=1;
% matrice temporaire des parametres
param=zeros(1,20);
% matrice globale des parametres
allparam=zeros(1,1);
allparam2=zeros(1,1);
% Création d'une matrice 2x2
M1=zeros(2,2);
dimlngcyto=0;

% Lecture des parametres cytométriques
% Le FLOWCAM produit un fichier de données contenant les valeurs maximales du
% signal de fluorescence et de phycoérithrine pour chacune des images
% que l'on retrouve dans les planches d'expérience ainsi que la largeur des signaux en
msec.

% On lit le fichier .fcm que l'on juxtapose pour former une matrice globale des
% parametres cytométriques représentée par la matrice M1.
%
for indfilefcm=1:length(datacyto)
   nomfilefcm=datacyto(indfilefcm,1).name;
   % lecture de la matrice elementaire
   M=dlmread(strcat(pathfilename,nomfilefcm),'\t',1,0);
   % Dimension de la matrice élémentaire
   [dim1,dim2]=size(M);
   [dimM1,dimM2]=size(M1);
   if dimM2==dim2
      % Collage de la matrice élémentaire a la matrice globale
      M1=[M1;M];
   else
      M1=M;
   end
end


% Décomposition des planches d'images tif en sous-images individuelles
% avec la fonction imcrop.
```

```
for indfiletiff=1:length(D)
  if indfiletiff > 0
    nomfichier=D(indfiletiff,1).name
    % lecture et rotation de la plachance images de 90 degrés
    % Cette rotation est nécessaire pour faire suivre les cellules
    % avec la matrice M1 des paramètres cytométriques.
    imgorig=imread(nomfichier)';

    % Recherche des objets de connectivité 8
    % Séparation des objets images : NUM1 représente la quantité d'objets
    % trouvés dans la planche image.
    % On sauvegarde sous forme de fichier .jpg chacun des objets qui seront
    % ensuite prétraités pour éliminer les informations inutiles.
    % BWLABEL retourne dans num1 le nombre d'images individuelles
    [L1,NUM1] = BWLABEL(imgorig,8);

    % Le parametre BoundingBox fournit par la fonction imfeature permet de connaitre la
position
    % de chacun des objet ainsi que leur dimension.

    resp1=imfeature(L1,'all');
    for i=1:NUM1
      % BoundingBox contient la position de l'objet sélectionné par i
      % imcrop conserve une portion d'image selon les coordonnées et les
      % dimensions de l'image ainsi que sa dimension en pixels sur la hauteur et largeur

      coord=resp1(i).BoundingBox;

        imgA2=imcrop(imgorig,coord);
        imgA3=histeq(imgA2); % Renormalisation de l'image par histogramme

        % Création de l'image originale avant traitement


%imwrite(imgA3,strcat(pathfilename,'imgorig_exp_',int2str(noexp),'_image_',int2str(i),ext
ension),'jpg');

        BW = im2bw(imgA3);  % Transformation en image binaire

        if size(BW) > 10
          % Détection des contours
          imgA5=edge(BW,'canny');

          % On applique un filtre
```

```
            H = [0,0,0;0,1,0;0,0,0];
            imgA3 = imfilter(imgA5,H);

            % Nettoyage de l'image à deux pixels des bordures

            [ligne,col]=size(imgA3);
            for lg=1:ligne
               for lc=1:col
                  if lg==2
                     imgA3(lg,lc)=0;
                  elseif lg==1
                     imgA3(lg,lc)=0;
                  elseif lg==ligne
                     imgA3(lg,lc)=0;
                  elseif lg==ligne-1
                     imgA3(lg,lc)=0;
                  elseif lc==1
                     imgA3(lg,lc)=0;
                  elseif lc==2
                     imgA3(lg,lc)=0;
                  elseif lc==col-1
                     imgA3(lg,lc)=0;
                  elseif lc==col
                     imgA3(lg,lc)=0;
                  end
               end
            end

            ESL90=strel('line',1,90);
            ESL0=strel('line',1,0);
            imgA5=imdilate(imgA3,[ESL90 ESL0]);

            H = [0,1,1;0,0,1;0,1,1];
            imgA5 = imfilter(imgA5,H);

            bwdfill=imgA5;
            bwnobord=imclearborder(bwdfill,8);
            imgtemp=bwnobord;

         %imshow(imgtemp);
% ------------------------------------------------------------------------
         % Extraction des parametres par imfeature et sauvegarde
         % dans la matrice temporaire param
% ------------------------------------------------------------------------
```

```
            [L,NUM] = BWLABELN(imgtemp,8);
          else
            NUM=-1;
          end
          if NUM > 0
            resp=imfeature(L,'all');
            qualiteimg=resp(1,1).EulerNumber;
            % paramètres d'images et paramètres cytométriques
            if qualiteimg == 0

              %imshow(resp(1,1).Image);
              param(1,1)=resp(1,1).Eccentricity; %
              param(1,2)=resp(1,1).Eccentricity/M1(nbcell,4)*1000; %
              param(1,3)=resp(1,1).MinorAxisLength/resp(1,1).MajorAxisLength;
              param(1,4)=M1(nbcell,4)/M1(nbcell,8);
              param(1,5)=resp(1,1).Centroid(1)-resp(1).Centroid(2);
              param(1,6)=resp(1,1).MajorAxisLength+resp(1,1).MinorAxisLength; %
M20+M02

param(1,7)=(resp(1,1).MajorAxisLength/resp(1,1).MinorAxisLength)*(M1(nbcell,4)/M1(n
bcell,8));
              param(1,8)=resp(1,1).MinorAxisLength;
              param(1,9)=M1(nbcell,4);
              param(1,10)=M1(nbcell,5);
              param(1,11)=M1(nbcell,8);
              param(1,12)=M1(nbcell,9);

param(1,13)=resp(1,1).MinorAxisLength/M1(nbcell,4)*resp(1,1).Centroid(1);
              param(1,14)=resp(1,1).MajorAxisLength/M1(nbcell,5);

param(1,15)=resp(1,1).MajorAxisLength/M1(nbcell,9)*resp(1,1).Centroid(1);
              param(1,16)=resp(1,1).Centroid(1)*M1(nbcell,4);
              param(1,17)=resp(1,1).MinorAxisLength/M1(nbcell,8);
              param(1,18)=noexp;
              param(1,19)=nbcell;
              param(1,20)=0;
            else
              param(1,1)=0;
              param(1,2)=0;
              param(1,3)=0;
              param(1,4)=0;
              param(1,5)=0;
              param(1,6)=0;
              param(1,7)=0;
```

```
        param(1,8)=0;
        param(1,9)=0;
        param(1,10)=0;
        param(1,11)=0;
        param(1,12)=0;
        param(1,13)=0;
        param(1,14)=0;
        param(1,15)=0;
        param(1,16)=0;
        param(1,17)=0;
        param(1,18)=noexp;
        param(1,19)=nbcell;
        param(1,20)=0;
    end

 else
      param(1,1)=0;
      param(1,2)=0;
      param(1,3)=0;
      param(1,4)=0;
      param(1,5)=0;
      param(1,6)=0;
      param(1,7)=0;
      param(1,8)=0;
      param(1,9)=0;
      param(1,10)=0;
      param(1,11)=0;
      param(1,12)=0;
      param(1,13)=0;
      param(1,14)=0;
      param(1,15)=0;
      param(1,16)=0;
      param(1,17)=0;
      param(1,18)=noexp;
      param(1,19)=nbcell;
      param(1,20)=0;

 end
 if size(allparam)==1
    allparam=param;
 else
    allparam=[allparam;param];
 end
 nbcell=nbcell+1;
```

```
      end
   end
   nbcell=1;
   noexp=noexp+1;

end
% Ajout de la matrice cible à la matrice allparam
% La matrice cible correspond à un tri visuelle sur deux espèces
% de phytoplancton.
% Dans ce cas particulier, c'est un tri visuelle sur Alexandrium tamarense.

%load cible;

%allparam=[allparam(:,1:14),cible];

% Nettoyage de la matrice allparam.mat
% Dépendant de la qualité des images, la forme de la cellule
% les parametres calculés
% par la fonction imfeature
nbcell=0;
for i=1:length(allparam)
   if allparam(i,1) > 0
      if allparam(i,2) > 0
         if nbcell==0
            nbcell=1;
            allparam2(1,1)=allparam(i,1);
            allparam2(1,2)=allparam(i,2);
            allparam2(1,3)=allparam(i,3);
            allparam2(1,4)=allparam(i,4);
            allparam2(1,5)=allparam(i,5);
            allparam2(1,6)=allparam(i,6);
            allparam2(1,7)=allparam(i,7);
            allparam2(1,8)=allparam(i,8);
            allparam2(1,9)=allparam(i,9);
            allparam2(1,10)=allparam(i,10);
            allparam2(1,11)=allparam(i,11);
            allparam2(1,12)=allparam(i,12);
            allparam2(1,13)=allparam(i,13);
            allparam2(1,14)=allparam(i,14);
            allparam2(1,15)=allparam(i,15);
            allparam2(1,16)=allparam(i,16);
            allparam2(1,17)=allparam(i,17);
            allparam2(1,18)=allparam(i,18);
            allparam2(1,19)=allparam(i,19);
```

```
            allparam2(1,20)=allparam(i,20);
          else
            allparam2=[allparam2;allparam(i,:)];
          end

      end
    end
end
% Sauvegarde de la matrice apres nettoyage
save allparam2;
% Sauvegarde de la matrice originale des parametres.
save allparam.mat
```

MODULE SEGM_CYTO.M

```
%
% ------------------------------------------------------
% ---------- SEGM_CYTO.M ----------------------
% ------------ Stratégie 2 ----------------------------
% ------------------------------------------------------
%
% Richard Lepage (Maitrise en ingénierie)
% Université du Québec à Rimouski
% Ce logiciel est concu selon des fonctions de MATLAB
%
% Librairie: network, image processing, statistic
%
% Lecture des paramètres nécessaires à la reconnaissance de la cellule
% selon les données paramétriques situées dans le fichier .fcm
%
%
%On efface le workspace et l'écran des résultats
clear;clc;
%-------------------------------------------------------
% ------ Lecture de la base de données no.1 -------
% ------qui contient des cellules apparentées ------
% -------------------------------------------------------

pathcytometre='c:/image_flowcam/database_1/*.fcm';
% datacyto contient la liste de tous les fichiers .fcm
datacyto=dir(pathcytometre);
% Chemin global de tous les fichiers
pathfilename='c:/image_flowcam/database_1/';
```

```
%Variable d'indice
nbcell=1;
noexp=1;
% matrice temporaire des parametres
param=zeros(1,20);
% matrice globale des parametres
allparam=zeros(1,1);
allparam1=zeros(1,1);
allparam2=zeros(1,1);
% matrice des informations cytométriques lues dans le fichier fcm
M1=zeros(2,2);
dimlngcyto=0;

% Lecture des parametres cytométriques
% Le FLOWCAM produit un fichier de données contenant les valeurs maximales du
% signal de fluorescence et de phycoérithrine pour chacune des images
% que l'on retrouve dans les planches d'expérience ainsi que la largeur
% des signaux en msec.

% On lit le fichier .fcm que l'on insere dans une matrice globale des
% parametres cytométriques représentée par la matrice M1.
%
for indfilefcm=1:length(datacyto)
   nomfilefcm=datacyto(indfilefcm,1).name;
   % lecture de la matrice elementaire
   M=dlmread(strcat(pathfilename,nomfilefcm),'\t',1,0);
   % Dimension de la matrice élémentaire
   [dim1,dim2]=size(M);
   [dimM1,dimM2]=size(M1);
   if dimM2==dim2
      % Collage de la matrice élémentaire a la matrice globale
      M1=[M1;M];
   else
      M1=M;
   end
end
[numl,numc]=size(M1);
% Pour chaque ligne de la matrice élémentaire, on lit les colonnes
% les plus significatives
datecyto=M1(nbcell,1);
noexp=1;
for i=1:numl
   if M1(nbcell,1) == datecyto
```

```
    else
       noexp=noexp+1;
       datecyto=M1(nbcell,1);
    end

    param(1,1)=M1(nbcell,3); %Cell area
    param(1,2)=M1(nbcell,4); % Chlorophyl peak
    param(1,3)=M1(nbcell,5); % Chlorophyl TOF
    param(1,4)=M1(nbcell,6); % Particules par chaines
    param(1,5)=M1(nbcell,7); % FFT area
    param(1,6)=M1(nbcell,8); % Phyco peak
    param(1,7)=M1(nbcell,9); % Phyco TOF
    param(1,8)=M1(nbcell,10); %Feret max diameter
    param(1,9)=M1(nbcell,11); % Feret min diameter
    param(1,10)=M1(nbcell,12); % Cell x
    param(1,11)=M1(nbcell,13); % Cell y
    param(1,12)=M1(nbcell,14); % Cell ESD
% Combinaison de parametres pour avoir une possibilité d'augmenter la
% dimension des parametres orthogonaux.
    param(1,13)=M1(nbcell,3)/M1(nbcell,7); % Cell Area/ FFT area
    param(1,14)=M1(nbcell,4)/M1(nbcell,5); % Chlorophyl peak/Chlorophyl TOF
    param(1,15)=M1(nbcell,8)/M1(nbcell,9); % Phyco peak/ Phyco TOF
    param(1,16)=M1(nbcell,3)/M1(nbcell,4); % Cell Area/Chlorophyl peak
    param(1,17)=M1(nbcell,5)/M1(nbcell,9); % Chloro TOF/Phyco TOF
    param(1,18)=noexp;
    param(1,19)=nbcell;
    param(1,20)=-1.0;
    if size(allparam)==1
       allparam=param;
    else
       allparam=[allparam;param];
    end
    nbcell=nbcell+1;
end
% ------------------------------------------------------------------------
% --------- Lecture de la base de données no.2 ----------------------------
% ----qui contient uniquement les cellules d'alexandrium tamarense -----
% ------------------------------------------------------------------------
% Fichier format texte des parametres cytométriques
pathcytometre='c:/image_flowcam/*.fcm';
% datacyto contient la liste de tous les fichiers .fcm
datacyto=dir(pathcytometre);
% Chemin global de tous les fichiers
pathfilename='c:/image_flowcam/';
```

```
%Variable d'indice
nbcell=1;
noexp=1;
dimlngcyto=0;
M1=zeros(2,2);
% Lecture des parametres cytométriques

for indfilefcm=1:length(datacyto)
   nomfilefcm=datacyto(indfilefcm,1).name;
   % lecture de la matrice elementaire
   M=dlmread(strcat(pathfilename,nomfilefcm),'\t',1,0);
   % Dimension de la matrice élémentaire
   [dim1,dim2]=size(M);
   [dimM1,dimM2]=size(M1);
   if dimM2==dim2
      % Collage de la matrice élémentaire a la matrice globale
      M1=[M1;M];
   else
      M1=M;
   end
end

[numl,numc]=size(M1);
datecyto=M1(nbcell,1);
noexp=1;
for i=1:numl
   if M1(nbcell,1) == datecyto
   else
      noexp=noexp+1;
      datecyto=M1(nbcell,1);
   end

   param(1,1)=M1(nbcell,3); %Cell area
   param(1,2)=M1(nbcell,4); % Chlorophyl peak
   param(1,3)=M1(nbcell,5); % Chlorophyl TOF
   param(1,4)=M1(nbcell,6); % Particules par chaines
   param(1,5)=M1(nbcell,7); % FFT area
   param(1,6)=M1(nbcell,8); % Phyco peak
   param(1,7)=M1(nbcell,9); % Phyco TOF
   param(1,8)=M1(nbcell,10); %Feret max diameter
   param(1,9)=M1(nbcell,11); % Feret min diameter
   param(1,10)=M1(nbcell,12); % Cell x
   param(1,11)=M1(nbcell,13); % Cell y
   param(1,12)=M1(nbcell,14); % Cell ESD
```

```
   param(1,13)=M1(nbcell,3)/M1(nbcell,7); % Cell Area/ FFT area
   param(1,14)=M1(nbcell,4)/M1(nbcell,5); % Chlorophyl peak/Chlorophyl TOF
   param(1,15)=M1(nbcell,8)/M1(nbcell,9); % Phyco peak/ Phyco TOF
   param(1,16)=M1(nbcell,3)/M1(nbcell,4); % Cell Area/Chlorophyl peak
   param(1,17)=M1(nbcell,5)/M1(nbcell,9); % Chloro TOF/Phyco TOF
   param(1,18)=noexp;
   param(1,19)=nbcell;
   param(1,20)=1.0;
   if size(allparam1)==1
      allparam1=param;
   else
      allparam1=[allparam1;param];
   end
   nbcell=nbcell+1;
end

% Sauvegarde de la matrice apres nettoyage
allparam2=[allparam;allparam1];
save allparam2;
```

Modules pour stratégie deux (2)

SANS DÉCORRÉLATION PAR LA MÉTHODE PCA

MODULE RANDOM_TRI_NOPCA.M

```
-----------------------------------------------
% -------- random_tri_nopca.m -------------------------
% -----------------------------------------------
% ---------- Strategie 1 et stratégie 2 ------
%
% Module de tri aléatoire permettant d'obtenir 33% des données de la base no.1 et no.2
% qui seront appliquées au réseau de neurones en tant que données d'apprentissage.
% Les données ne sont pas traitées par la méthode PCA.
% Richard Lepage
% Étudiant Maitrise en ingénierie
% 2004
% -----------------------------------------------
%
% On efface l'écran et les variables précédentes
clear;clc;
delete net.mat;
% Chargement de la matrice globale des paramètres du fichier fcm obtenus
% a partir du module segm_cyto.m

load allparam2 allparam2;
[rowmax,colmax]=size(allparam2);

% On charge les 17 premieres colonnes de la matrice allparam2 qui compte
% 20 colonnes dans la matrice AllParamBeforePCA.
AllParamBeforePCA=allparam2;
% On crée une table qui contiendra les
% objets cibles.
classe1=zeros(1,1);

% Tirage aléatoire de 33% de la matrice d'apprentissage
[rowcl,colcl]=size(AllParamBeforePCA);
R=round(rand(round((rowcl-1)*0.33),1)*rowcl);

% On lit la table AllParamBeforePCA selon la valeur de nrandom et on
% insere dans la table classe1 les classes numérotés
```

```
% targetclasse contient les classes cibles pour chacune des images.
for i=1:size(R)
      nrandom=R(i);
      if nrandom < rowcl+1
         if nrandom > 0
            if size(classe1)==1
               classe1=AllParamBeforePCA(nrandom,:);

            else
               classe1=[classe1;AllParamBeforePCA(nrandom,:)];

            end
         end
      end
end

targetclasse=classe1(:,20);
% Sauvegarde de la matrice d'apprentissage en fichier .mat
save classe1.mat;

% Sauvegarde des matrices de parametres de la méthode PCA
save targetclasse;
save colmax;
% Fin de ce module.
```

APPRENTISSAGE_CYTO_NOPCA.M

```
% ------------------------------------------------------
% -------- apprentissage_cyto_nopca.m -----------
% ------------------------------------------------------
% ---------- Stratégie 2 ---------------------------
%
% Module d'apprentissage sans décorrélation par la méthode PCA.
% Richard Lepage
% Étudiant Maitrise en ingénierie
% 2004
% ------------------------------------------------
%
% On efface l'écran et les variables précédentes
clear;clc;
load classe1;
load colmax;
load targetclasse;
```

```
% allclasse peut contenir plusieurs classes.
%allclasse=[classe1;classe2;classe3,....,classeN]';
% Dans allclasse, on retrouve seulement 33% des données d'apprentissage
% choisi aléatoirement selon une courbe normale dans la base de données
% no.1 et no.2.

allclasse=[classe1]';

% La fonction d'activation utilisée pour les couches cachées est tansig tandis que
% la fonction d'activation purelin se retrouve dans la couche d'entrée
% et la couche de sortie.

% Ce programme utilise la fonction NEWFF pour la création d'un réseau MLP.
% Trainbfg est l'algorithme de convergence quasi-Newton.

% ------------------------------------------------
% Création du réseau de neurone.
neuralinput=zeros(1,1);
for ninput=1:colmax-3
   if ninput==1
      neuralinput=[-1 1];
   else
      neuralinput=[neuralinput;[-1 1]];
   end
end
net=newff(neuralinput,[36,18,1],{'tansig','tansig','purelin'},'trainbfg');
% ------------------------------------------------

% Initialisation aléatoire des poids du réseau.
net=init(net);
% Méthode de régularisation de la moyenne au carré de l'erreur.
net.performFcn='msereg';
% Initialisation des paramètres d'apprentissage du réseau.
net.adaptParam.passes=1;
net.trainparam.show=5;
net.trainparam.lr=0.001;
net.trainparam.epochs=1000;
net.trainparam.goal=1e-6;
net.trainparam.mu=.001;

% allclasse contient tous les parametres moins les trois dernières colonnes
allclasse=allclasse(1:colmax-3,:);
% Entrainement du réseau de neurones avec les données d'apprentissage
```

```
[net,tr]=train(net,allclasse,targetclasse');
% Sauvegarde de la structure du réseau pour l'étape de classification
save net;
target=allparam2(:,20);
save target;
% Fin de ce module.
```

MODULE DE CLASSIFICATION.

```
% ------------------------------------------
% ---------- detection_strategie_2_nopca.M ----
% ---------- Stratégie 2 ----------
% ------------------------------------------
%
% Richard Lepage (Maitrise en ingénierie)
% Université du Québec à Rimouski
% Ce logiciel est concu selon des fonctions de MATLAB
%
% Librairie: network, image processing, statistic
%
% Lecture des paramètres nécessaires à la reconnaissance de la cellule
% selon les données paramétriques situées dans le fichier .fcm
%
%
%On efface le workspace et l'écran des résultats
clear;clc;
% Fichier format texte des parametres cytométriques
pathcytometre='c:/image_flowcam/database_1/*.fcm';
% datacyto contient la liste de tous les fichiers .fcm
datacyto=dir(pathcytometre);
% Chemin global de tous les fichiers
pathfilename='c:/image_flowcam/database_1/';
%Variable d'indice
nbcell=1;
noexp=1;
% matrice temporaire des parametres
param=zeros(1,20);
% matrice globale des parametres
allparam=zeros(1,1);
allparam2=zeros(1,1);
% Création d'une matrice 2x2
M1=zeros(2,2);
dimlngcyto=0;
```

```
% Lecture des parametres cytométriques
% Le FLOWCAM produit un fichier de données contenant les valeurs maximales du
% signal de fluorescence et de phycoérithrine pour chacune des images
% que l'on retrouve dans les planches d'expérience ainsi que la largeur des signaux en
msec.

% On lit le fichier .fcm que l'on insere dans une matrice globale des
% parametres cytométriques représentée par la matrice M1.
%
for indfilefcm=1:length(datacyto)
   nomfilefcm=datacyto(indfilefcm,1).name;
   % lecture de la matrice elementaire
   M=dlmread(strcat(pathfilename,nomfilefcm),'\t',1,0);
   % Dimension de la matrice élémentaire
   [dim1,dim2]=size(M);
   [dimM1,dimM2]=size(M1);
   if dimM2==dim2
      % Collage de la matrice élémentaire a la matrice globale
      M1=[M1;M];
   else
      M1=M;
   end
end
[numl,numc]=size(M1);
datecyto=M1(nbcell,1);
noexp=1;
for i=1:numl
   if M1(nbcell,1) == datecyto
   else
      noexp=noexp+1;
      datecyto=M1(nbcell,1);
   end
   param(1,1)=M1(nbcell,3); %Cell area
   param(1,2)=M1(nbcell,4); % Chlorophyl peak
   param(1,3)=M1(nbcell,5); % Chlorophyl TOF
   param(1,4)=M1(nbcell,6); % Particules par chaines
   param(1,5)=M1(nbcell,7); % FFT area
   param(1,6)=M1(nbcell,8); % Phyco peak
   param(1,7)=M1(nbcell,9); % Phyco TOF
   param(1,8)=M1(nbcell,10); %Feret max diameter
   param(1,9)=M1(nbcell,11); % Feret min diameter
   param(1,10)=M1(nbcell,12); % Cell x
   param(1,11)=M1(nbcell,13); % Cell y
```

```
    param(1,12)=M1(nbcell,14); % Cell ESD
    param(1,13)=M1(nbcell,3)/M1(nbcell,7); % Cell Area/ FFT area
    param(1,14)=M1(nbcell,4)/M1(nbcell,5); % Chlorophyl peak/Chlorophyl TOF
    param(1,15)=M1(nbcell,8)/M1(nbcell,9); % Phyco peak/ Phyco TOF
    param(1,16)=M1(nbcell,3)/M1(nbcell,4); % Cell Area/Chlorophyl peak
    param(1,17)=M1(nbcell,5)/M1(nbcell,9); % Chloro TOF/Phyco TOF
    param(1,18)=noexp;
    param(1,19)=nbcell;
    param(1,20)=1.0;
    if size(allparam)==1
       allparam=param;
    else
       allparam=[allparam;param];
    end
    nbcell=nbcell+1;
end
% Sauvegarde de la matrice apres nettoyage
allparam2=allparam;

% Chargement de la matrice des poids préalablement calculée
% par le module apprentissage_cyto_nopca.m
load net.mat net;
allparam2beforepca=allparam2(:,1:17)';
delete resultat_perf.mat;
% Mode de reconnaissance par simulation
% Classification des données cytométriques de la base de données no.1 ou no.2
resultat=sim(net,allparam2beforepca);
[lngmax,colmax]=size(resultat);
% On compte le nombre de bonne classification.
compteur=0;
for indicecol=1:colmax
   res_a=resultat(1,indicecol);
   if res_a>0.9
      compteur=compteur+1;
   end
end
compteur
```

Module de performance sur l'apprentissage du réseau de neurones.

PERF_CLASSE_NOPCA.M

```
% ------------------------------------------------
% -------- perf_classe_nopca.m -------------------------
% ------------------------------------------------
% ---------------- Strategie 1 et stratégie 2 ------
%% Programme de classification
% Les valeurs des poids du réseau de
% neurones ont été calculés préalablement lors de l'étape d'apprentissage.
clc;clear;
% Chargement de la matrice des poids
load net.mat;
% Chargement de la matrice des données produite
% par le module segm.m ou segm_cyto.m dépendant de la stratégie
load allparam2.mat;
load target target;

allparam2beforepca=allparam2(:,1:17)';

delete resultat_perf.mat;
% Présentation des données au réseau de neurone en mode simulation
% Ce mode permet de classer les données selon la valeur calculée des poids
% du réseau de neurones.
resultat=sim(net,allparam2beforepca);

resultat_perf=[resultat;target'];

[lngmax,colmax]=size(resultat_perf);
% On compte le nombre de bonne classification.
compteur=0;
for indicecol=1:colmax
   res_a=resultat_perf(1,indicecol);
   res_cible=double(resultat_perf(2,indicecol));
   if res_a > .9
      compteur=compteur+1;

   end
end
compteur
save resultat_perf.mat;
```

Modules pour la stratégie un (1)

Module d'apprentissage

```
%
% ------------------------------------------------
% ---------- Module: apprentissage.m ---------------
% ------------------------------------------------
%
% Module d'apprentissage des formes présélectionnées.
% Richard Lepage
% Étudiant Maitrise en ingénierie
% 2004
clear;clc;
delete net.mat;
% Chargement de la matrice globale des paramètres
load allparam2;

% Liste des paramètres pour chaque colonne
% colonne 1:  Excentricité de l'objet
% colonne 2:  Excentricité/chlorophyl peak
% colonne 3:  Longueur de l'axe mineur/ Longueur de l'axe majeur
% colonne 4:  Chloro peak/ Phyco peak
% colonne 5:  centroide x - centroide y
% colonne 6:  Longueur de l'axe majeur + longueur de l'axe mineur
% colonne 7:  (Longueur de l'axe majeur/longueur de l'axe mineur)*(chloro peak/phyco
peak)
% colonne 8:  Longueur de l'axe mineur
% colonne 9:  Chloro peak
% colonne 10:  Chloro TOF
% colonne 11:  Phyco peak
% colonne 12:  Phyco TOF
% colonne 13:  Numéro de la planche
% colonne 14:  Indice de l'image de la cellule
% colonne 15:  Valeur cible utilisé lors de l'apprentissage de la classe

% Utilisation de la méthode PCA pour éliminer tout types de
% corrélation dans les dimensions utilisées.

% On charge les 12 premieres colonnes de la matrice allparam qui compte
% 15 colonnes.
```

```
AllParamBeforePCA=allparam2(:,1:17)';
% La fonction prestd permet de centrer les données autour d'une moyenne nulle et
% calcule un écart type égale à un.
[pn,meanp,stdp] = prestd(AllParamBeforePCA);
% Application de la méthode PCA pour éliminer les corrélations
[ptrans,transMat] = prepca(pn,0.04);
% Normalisation des données d'entrées entre -1 et +1
[AllParamAfterPCA,minp,maxp] = premnmx(ptrans);

% On ramène les colonnes 18 à 20 dans la matrice AllParamAfterPCA
AllParamIndice=allparam2(:,18:20)';
AllParamAfterPCA=[AllParamAfterPCA;AllParamIndice]';

% Liste des cellules trié selon la qualité de l'image
% on lit la table allparamAfterPCA pour séparer les deux classes
% qui serviront pour l'apprentissage du réseau de neurones
[lngmax,colmax]=size(AllParamAfterPCA);
% On crée deux tables qui contiendront les
% objets cibles.
classe1=zeros(1,1);

% On parcourt toute la table AllParamAfterPCA, ligne apres ligne et on
% insere dans la table appropriée les classes numérotés
for i=1:lngmax
   if AllParamAfterPCA(i,colmax)==1
      if size(classe1)==1
         classe1=AllParamAfterPCA(i,:);
      else
         classe1=[classe1;AllParamAfterPCA(i,:)];
      end
   end
end
% Sauvegarde des deux matrices en fichier .mat
save classe1.mat

% allclasse peut contenir plusieurs classes.
%allclasse=[classe1;classe2;classe3,....,classeN]';
allclasse=[classe1]';

% La fonction d'activation pour les couches cachées est tansig tandis que
% la fonction d'activation purelin est utilisée pour la couche d'entrée
% et la couche de sortie.

% Ce programme utilise la fonction NEWFF pour la création d'un réseau MLP.
```

```
% Création du réseau de neurone.
% Configuration du réseau de neurones:
% 10 neurones en entrées (10 paramètres)
% xx neurones première couche cachée
% xx neurones seconde couche cachée
% Un neurone dams la couche de sortie (-1, 0 +1)
neuralinput=zeros(1,1);
for ninput=1:colmax-3
   if ninput==1
      neuralinput=[-1 1];
   else
      neuralinput=[neuralinput;[-1 1]];
   end
end

net=newff(neuralinput,[16,8,1],{'tansig','tansig','purelin'},'trainlm');
%
% Initialisation aléatoire des poids du réseau.
net=init(net);
%net.layers{1}.initFcn='initnw';
%net.layers{2}.initFcn='initnw';
net.adaptParam.passes = 1;
% Initialisation des paramètres d'apprentissage du réseau.
%net.trainFcn='trainr';
net.trainparam.show=5;
net.trainparam.epochs=5000;
net.trainparam.goal=1e-10;
net.trainparam.mu=.001;
% targetclasse contient les classes cibles pour chacune des images.
% targetclasse contient uniquement la ligne 11
targetclasse=allclasse(colmax,:);
% allclasse contient tous les parametres moins les trois dernières colonnes
allclasse=allclasse(1:colmax-3,:);

[net,tr]=train(net,allclasse,targetclasse);
save net;
save allclasse.mat;
save targetclasse.mat;
% Fin de ce module.
```

Programme de classification

```
% Programme de classification
% ---------------------------------------------
% --------- classification.m ------------------
% ---------------------------------------------
%
% La valeur des poids du réseau de
% neurones ont été calculés préalablement lors de l'étape d'apprentissage.
clc;clear;
% Chargement de la matrice des poids
load net.mat;
% Chargement de la matrice des planches images obtenue
% a l'aide du programme segm.m et nettoyées des images superflues
load allparam2.mat;
allcellules=allparam2(:,1:17)';
% Centrage des données moyenne=0 std=1
[pn,meanp,stdp] = prestd(allcellules);
% Décorrélation par la méthode PCA
[ptrans,transMat] = prepca(pn,0.04);
% Normalisation des données d'entrées entre -1 et +1
[pnnorm,minp,maxp] = premnmx(ptrans);
% Application des 8 parametres au réseau de neurones
resultat=sim(net,pnnorm);
% Résultat en sortie du réseau de neurones retransmis a la
% colonne 15 de la matrice allparam.mat
resultat=[resultat;allparam2(:,20)';allparam2(:,18)';allparam2(:,19)'];

[lngmax,colmax]=size(resultat);
% nettoyage des résultats selon un seuil prédéfini.
for i=1:colmax
   if resultat(1,i) > 0.9999
      resultat(1,i)=1;
   elseif resultat(1,i)<-0.9999
      resultat(1,i)=-1;
   else
      resultat(1,i)=0;
   end
end
save resultat.mat
```

Simulation d'un neurone formel en code MATLAB

```
% Un neurone simple avec deux entrée et une sortie.
% La fonction d'activation est linéaire

% On efface tous les objets
clear;

% Seuil des entrées
R1=[-1 1;-1 1];

% Création d'un neurone avec deux entrées
net=newlin(R1,1);

% Initialisation w(1)=1 et w(2)=2
net.iw{1,1}=[1 2];

% Initialisation du biais b=0
net.b{1}=1;

% Vecteur d'entrée
p1=[-1 2 -2 3 0] ;
p2=[-2 1 -3 1 0]
P1=[p1;p2];

% Calcul de la sortie sans modification des poids
Y=sim(net,P1);

% Matrice représentant les données cible qui servent de professeur
T=[-1 1 -1 1 -1]

for i=1:1:length(p1)
   if T(i)==-1
      plot(p1(i),p2(i),'X');
   else
      plot(p1(i),p2(i),'O');
   end
   if i==1
      hold;
   end
end
x=min(p1):.1:max(p1);
```

```
w3=net.b{1};
w=net.iw{1,1};
w1=w(1);w2=w(2);
plot(x,-(w3/w2)-(w1/w2)*x);

% Apprentissage du réseau
net.inputWeights{1,1}.learnParam.lr=0.1;
net.biases{1}.learnParam.lr=0.1;
net.trainParam.epochs=4;
net=train(net,P1,T);
x=min(p1):.1:max(p1);
w3=net.b{1};
w=net.iw{1,1};
w1=w(1);w2=w(2);
        plot(x,-(w3/w2)-(w1/w2)*x);
```

Code MATLAB d'un réseau SOM

Le programme MATLAB ci-dessous est la simulation d'un réseau SOM sur des données fictives représentant deux nuages de points.

```
%% Démonstration d'une carte auto-organisatrice a deux dimensions
%-------------------------------------------------
% Cette carte auto-organisatrice apprendra a représenter différentes régions
% selon les vecteurs présentés à l'entrée.
% Dans cet exemple, les neurones s'organise eux-mêmes dans une grille à deux dimensions
% plutôt qu'à une dimension.
%%
clear all;
clc;
load nuages_1.txt;
load nuages_2.txt;
load nuages_3.txt;
load nuages_4.txt;
load nuages_5.txt;
load nuages_6.txt;

% Apprentissage sur les données du nuages spécifié

[m n]=size(nuages_1);
P=[premnmx(nuages_1(:,1)');premnmx(nuages_1(:,2)')];
% Prétraitement des données d'entrées
T=hardlims(nuages_1(:,3))';
P1=[P(1,1:1:m);P(2,1:1:m)];
T1=[T(1,1:1:m)];
% Affichage du nuage de points.
figure(1);
subplot(2,2,1);
plotpv(P,hardlim(T));
xlabel('Données originales');
%%

figure(1);
subplot(2,2,2);
plotpv(P1,hardlim(T1));
xlabel('Donnees apprentissage');
```

```
% On crée le réseau compétitif de dimension 6x4, i.e 24 neurone
net = newsom(minmax(P1),[6 4]);

% Entrainement du réseau
net.trainParam.epochs = 1;
net = train(net,P1);

% Affichage de la carte auto-organisatrice
figure(1);
subplot(2,2,3);
plotsom(net.iw{1,1},net.layers{1}.distances);

%%
% On simule maintenant pour l'ensemble des données

% Simulation nuages_1
s = sim(net,P);
s1=hardlims(premnmx(vec2ind(s)));
figure(1);
subplot(2,2,4);
plotpv(P,hardlim(s1));
malclasse=0;
for i=1:1:m
   if T(1,i) == s1(i)
   else
      malclasse=malclasse+1;
   end
end
pctage=100-(malclasse/m)*100

% Simulation sur nuages_2.txt
[m n]=size(nuages_2);
P=[nuages_2(:,1)';nuages_2(:,2)'];
% Prétraitement des données d'entrées
T=hardlim(nuages_2(:,3))';
s = sim(net,P);
s1=hardlim(premnmx(vec2ind(s)));
malclasse=0;
for i=1:1:m
   if T(1,i) == s1(1,i)
   else
      malclasse=malclasse+1;
   end
```

```
end
pctage=100-(malclasse/m)*100

% Simulation sur nuages_3.txt
[m n]=size(nuages_3);
P=[nuages_3(:,1)';nuages_3(:,2)'];
% Prétraitement des données d'entrées
T=hardlim(nuages_3(:,3))';
s = sim(net,P);
s1=hardlim(premnmx(vec2ind(s)));
malclasse=0;
for i=1:1:m
   if T(1,i) == s1(1,i)
   else
      malclasse=malclasse+1;
   end
end
pctage=100-(malclasse/m)*100

% Simulation sur nuages_4.txt
[m n]=size(nuages_4);
P=[nuages_4(:,1)';nuages_4(:,2)'];
% Prétraitement des données d'entrées
T=hardlim(nuages_4(:,3))';
s = sim(net,P);
s1=hardlim(premnmx(vec2ind(s)));
malclasse=0;
for i=1:1:m
   if T(1,i) == s1(1,i)
   else
      malclasse=malclasse+1;
   end
end
pctage=100-(malclasse/m)*100

% Simulation sur nuages_5.txt
[m n]=size(nuages_5);
P=[nuages_5(:,1)';nuages_5(:,2)'];
% Prétraitement des données d'entrées
T=hardlim(nuages_5(:,3))';
s = sim(net,P);
s1=hardlim(premnmx(vec2ind(s)));
malclasse=0;
for i=1:1:m
```

```
   if T(1,i) == s1(1,i)
   else
      malclasse=malclasse+1;
   end
end
pctage=100-(malclasse/m)*100

% Simulation sur nuages_6.txt
[m n]=size(nuages_6);
P=[nuages_6(:,1)';nuages_6(:,2)'];
% Prétraitement des données d'entrées
T=hardlim(nuages_6(:,3))';
s = sim(net,P);
s1=hardlim(premnmx(vec2ind(s)));
malclasse=0;
for i=1:1:m
   if T(1,i) == s1(1,i)
   else
      malclasse=malclasse+1;
   end
end
pctage=100-(malclasse/m)*100
```

Les figures A-1 à A-6 donnent une interprétation visuelle de la classification effectuée par un réseau compétitif. L'apprentissage a été fait selon le nuage de points nuages_1.txt. L'apprentissage est assez long parce que nous devons utiliser la valeur de 500 itérations au minimum pour obtenir un bon taux de vraisemblance au niveau de la classification.

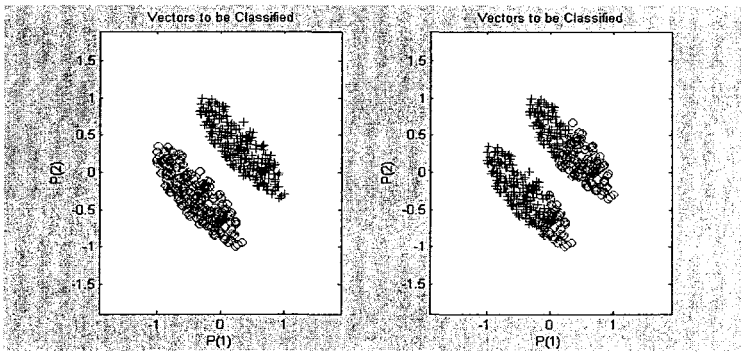


Figure A-1 Apprentissage compétitif (Nuage A)

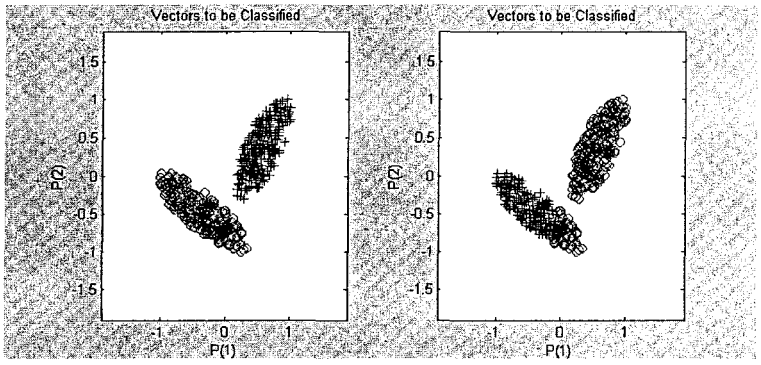


Figure A-2 Apprentissage compétitif (nuage B)

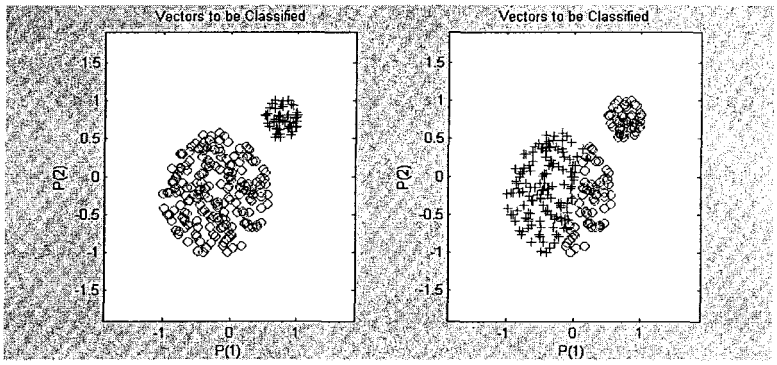


Figure A-3 Apprentissage compétitif (Nuage-C)

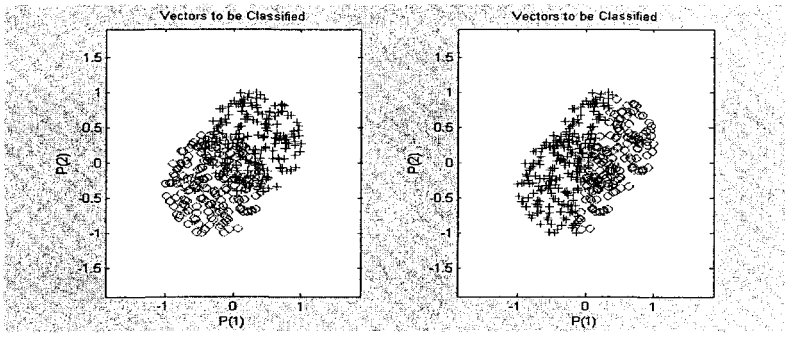


Figure A-4 Apprentissage compétitif (Nuage-D)

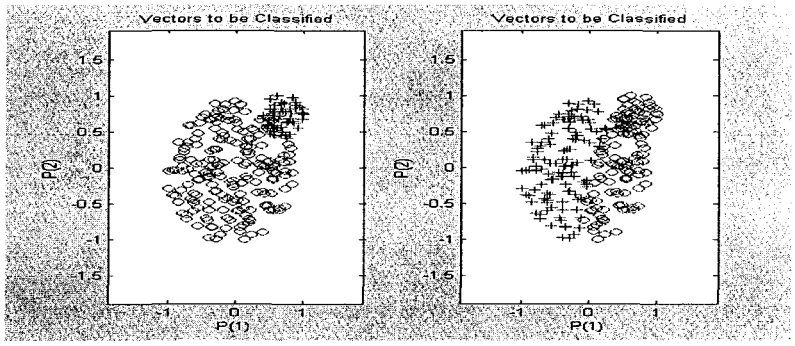


Figure A-5 Apprentissage compétitif (Nuage-E)

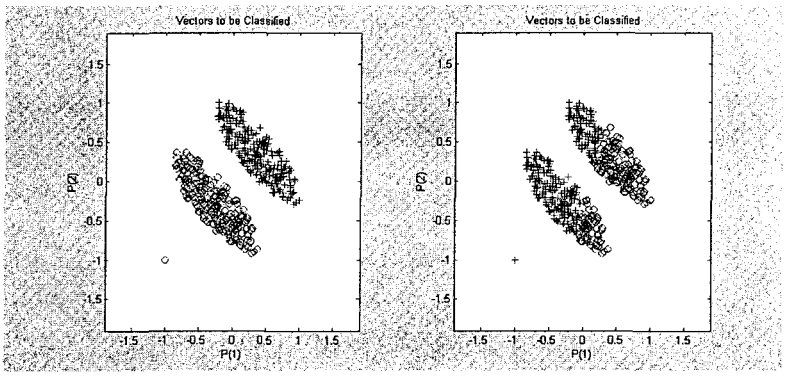


Figure A-6 Apprentissage compétitif (Nuage-F)

Discrimination de signaux par réseau MLP

Ce programme permet de discriminer des patrons de signaux qui sont cachées dans un signal présenté à l'entrée du réseau. Une fenêtre contenant 10 points ou vecteurs circule en continu dans le signal. Le patron représenté par ces 10 points est ensuite classifié par le réseau MLP. Les patrons sont composés d'un signal pur. Pour fin de généralisation, on peut bruiter chacun des patrons avec un bruit blanc pour ensuite les mémoriser dans un réseau MLP. De cette manière, un patron moindrement perturbé sera reconnu parmi les autres selon une certaine distance ou seuil permettant de glisser vers le patron le plus proche qui lui ressemble (phénomène de généralisation).

```
%% Classification de signaux par réseau MLP à une couches cachée de 3 neurones et
% une couche de sortie composée de 1 neurone.
% La fonction d'activation pour la couche cachée est tansig tandis que
% la fonction d'activation purelin est utilisée pour la couche de sortie.

% Ce programme utilise la fonction NEWFF pour la création d'un réseau MLP.
% qui mémorise les patrons de signaux que l'on peut trouver dans le signal de référence.
% Definition de 9 signaux et des cibles associées T.

clear;clc;
% Chargement du signal de référence.
load signal_ref.txt;
% Chargement de la matrice du patron des signaux.
load patrons.txt;

S1=premnmx(patrons(:,1)');
S2=premnmx(patrons(:,2)');
S3=premnmx(patrons(:,3)');
S4=premnmx(patrons(:,4)');
S5=premnmx(patrons(:,5)');
S6=premnmx(patrons(:,6)');
S7=premnmx(patrons(:,7)');
S8=premnmx(patrons(:,8)');
```

```
S9=premnmx(patrons(:,9)');

patrons=[S1;S2;S3;S4;S5;S6;S7;S8;S9]';
T=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9];
% On a besoin de l'information nligne pour connaitre la longueur du signal
% de référence.
[nligne ncol]=size(signal_ref);
% Matrice contenant le compte des patrons reconnues dans le signal de
% référence.
tdata=zeros(1,9);
%AFFSIG
% Création du réseau de neurone.
net=newff([-1 1;-1 1;-1 1;-1 1;-1 1;-1 1;-1 1;-1 1;-1 1;-1
1],[5,1],{'tansig','purelin'},'trainlm');
%
% Initialisation aléatoire des poids du réseau.
net=init(net);
% Initialisation des paramètres d'apprentissage du réseau.
net.trainparam.show=5;
net.trainparam.epochs=10;
net.trainparam.goal=1e-10;
% Entrainement du réseau à reconnaitre les patrons de signaux.
[net,tr]=train(net,patrons,T);
% Dans cette section, on déplace une fenetre de 10 échantillons par step de
% 1 échantillon.
for i=1:1:nligne
   if i < nligne-9
      % on lit les 10 échantillons (fenetre d'échantillons)
      source=signal_ref(i:1:9+i,1);
      % On prétraite les données d'entrées pour les normaliser.
      p=premnmx(source);
      % Simulation du réseau pour connaitre la valeur de sortie
      a=sim(net,p);
      % On converti en chaine de caractère la valeur de sortie du réseau
      % pour la comparer aux valeurs cible.
      TT1=NUM2STR(a);
      % Correspondant à la valeur cible, on incrémente la matrice des
      % resultat de confidence.
      switch(TT1)
         case '0.1',
            tdata(1,1)=tdata(1,1)+1;
         case '0.2',
            tdata(1,2)=tdata(1,2)+1;
         case '0.3',
```

```
            tdata(1,3)=tdata(1,3)+1;
        case '0.4',
            tdata(1,4)=tdata(1,4)+1;
        case '0.5',
            tdata(1,5)=tdata(1,5)+1;
        case '0.6',
            tdata(1,6)=tdata(1,6)+1;
        case '0.7',
            tdata(1,7)=tdata(1,7)+1;
        case '0.8',
            tdata(1,8)=tdata(1,8)+1;
        case '0.9',
            tdata(1,9)=tdata(1,9)+1;
      end
    else
      % Si on atteint la fin des données du signal de référence,
      % On sort de la boucle for.
      break;
    end
end

% Fin du programme.
```

Processus de segmentation d'une image

Ce programme permet de segmenter une image qui contient plusieurs objets. Les objets présentant des points de contacts avec la bordure de l'image sont éliminés pour ne garder que l'objet principal de l'image.

```
Programme de segmentation d'une image
% Remarque: Le signe % est un commentaire.
%% Lecture de l'image
img=imread('psdn.jpg');
%figure,imshow(img),title('Image originale');
%% Conversion d'une image RGB en image a teinte de gris
imgray=rgb2gray(img);
%figure,imshow(imgray),title('Image a teinte de gris');
%% Détection des cellules avec filtre de gradient binaire de Sobel
bw1=edge(imgray,'sobel',(graythresh(imgray)*.1));
%figure,imshow(bw1),title('filtre a gradient binaire de Sobel');
%% Linéarisé les contour pour enlever les discontinuités
ESL90=strel('line',3,90);
ESL0=strel('line',3,0);
bwdil=imdilate(bw1,[ESL90 ESL0]);
%figure,imshow(bwdil),title('Image dilate par élément structurel linéaire');
%% Remplir les espaces vides a l'intérieur du contour
bwdfill=imfill(bwdil,'holes');
%figure,imshow(bwdfill),title('Image binaire avec remplissage des vides internes');
%% Élimination des objets qui font parties de la bordure de l'image
bwnobord=imclearborder(bwdfill,4);
%figure,imshow(bwnobord),title('Image dilatée par élément structurel linéaire');
%% Adoucissement de l'objet
seD=strel('diamond',1);
BWfin=imerode(bwnobord,seD);
BWfin=imerode(BWfin,seD);
%figure,imshow(BWfin),title('Image segmentée');
%% Affichage du contour de l'objet isolé.
BWoutline=bwperim(BWfin);
Segout=imgray;
Segout(BWoutline)=255;
figure,imshow(BWoutline);
```

Code MATLAB d'un réseau Bayésien

Le programme suivant permet d'approcher une fonction théorique et compare avec diverses fonctions qui ont été bruitées pour tester la capacité de généralisation d'un réseau RBF.

```
%% Approximation de fonctions par réseau à base radiale
%
% Ce programme utilise la fonction NEWRB pour la création d'un réseau à base radiale
% qui approxime une fonction définies par un ensemble de points.
%
% Ce programme est une version modifiée d'un programme
% que l'on retrouve dans la section démo du logiciel MATLAB.
%
%%
% Definition de 21 entrée P et des cibles associées T.

clear;clc;
load fonction.txt;
% L'ensemble des données
X1=premnmx(fonction(:,1)');
X2=premnmx(fonction(:,2)');
X3=premnmx(fonction(:,3)');
t1=1:9:51;
% Sous-ensemble correspondant aux données d'apprentissage pour le réseau
% de neurones RBF (Radial Basis Functions).
% Courbe de sortie théorique
e1=X1(t1);
e2=X2(t1);
e3=X3(t1);

% L'ensemble des sorties connues pour le signal mesuré.
T=fonction(:,9)';
% pré-traitement des données échelle -1..1
T=premnmx(T);
% Sous-ensemble des sorties connues.
target=T(t1);

[M,N]=size(target);
```

```
P =[e1;e2;e3];
% La fonction NEWRB crée rapidement un réseau à base radiale qui approxime
% la fonction définie par P et T.  En supplément des paramètres d'apprentissage et de cible,
% NEWRB prends deux autres arguments, soit seuil du carré de la somme de l'erreurs
% et la constante de propagation.
eg = 0.02; % seuil du carré de la somme de l'erreurs
sc = 1;    % constante de propragation
net = newrb(P,target,eg,sc);

%%
% Pour voir comment le réseau se comporte, on trace de nouveau l'ensemble des vecteur
% d'apprentissage. On simule alors le réseau avec la fonction sim pour des
% entrées ayant la meme échelle.



P=[X1;X2;X3];
Y = sim(net,P);
subplot(2,3,1);plot(t1,target,'+');
title('Fig. 1 Vect apprentissage');
subplot(2,3,2);plot(1:1:51,Y,'b.');
title('Fig.2 Approx. fonction');
hold;
plot(1:1:51,T);
subplot(2,3,3);
axis([0 1 0 1]);
text(.05,.9,strcat('Moy (T):',num2str(mean(T))))
text(.05,.8,strcat('var (T):',num2str(std(T))))
text(.05,.7,strcat('Moy (Y):',num2str(mean(Y))))
text(.05,.6,strcat('var (Y):',num2str(std(Y))))

%sprintf('Moy (Y):%f - Variance :%f',mean(Y),std(Y))

% On utilise toute les données pour l'apprentissage
P =[X1;X2;X3];
eg = 0.02; % seuil du carré de la somme de l'erreurs
sc = 1;    % constante de propragation
net = newrb(P,T,eg,sc);
% On simule avec la totalité des vecteurs d'entrées.
P=[X1;X2;X3];
Y = sim(net,P);
figure(1);
subplot(2,3,4);plot(1:1:51,T,'+');
title('Fig.3 Vect apprentissage');
```

```
subplot(2,3,5);plot(1:1:51,Y,'b.');
hold;
plot(1:1:51,T);
title('Fig.4 Approx. fonction');
subplot(2,3,6);
axis([0 1 0 1]);
text(.05,.9,strcat('Moy (T):',num2str(mean(T))))
text(.05,.8,strcat('var (T):',num2str(std(T))))
text(.05,.7,strcat('Moy (Y):',num2str(mean(Y))))
text(.05,.6,strcat('var (Y):',num2str(std(Y))))
```

Sur les six (6) graphiques, on retrouve la forme d'onde théorique ainsi que la forme d'onde mesurée ou réelle. Sur les autres graphiques, on retrouve la forme d'onde mesurée associée à différents bruits. Nous devons, par généralisation, vérifier si les formes d'onde bruitées approchent la forme d'onde théorique que l'on a fait apprendre par le réseau Bayésien.

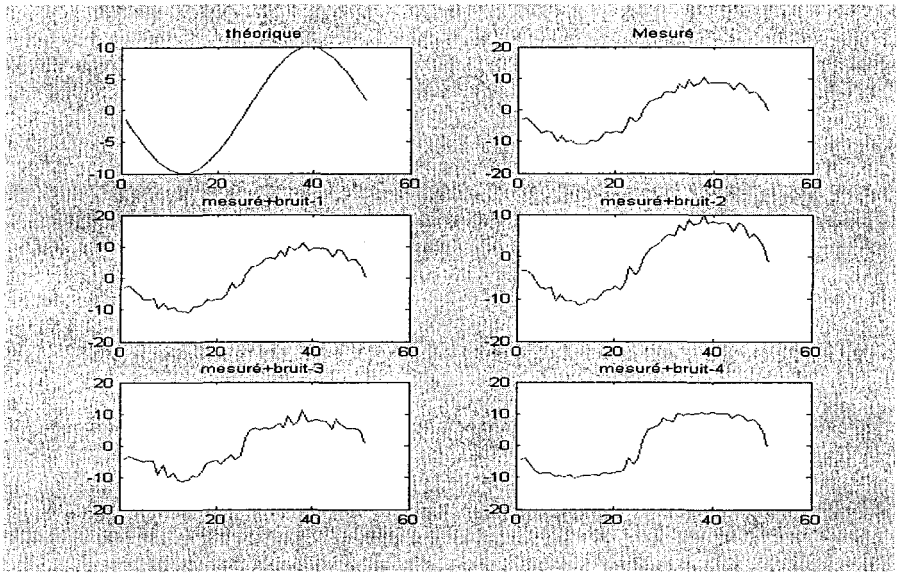


Figure A-7 Formes d'ondes bruitées

La figure A-8 représente tous les signaux qui doivent être généralisés. Selon des données présentées à l'entrée du réseau, on vérifie comment se comportera le réseau au niveau de l'approximation d'une fonction.

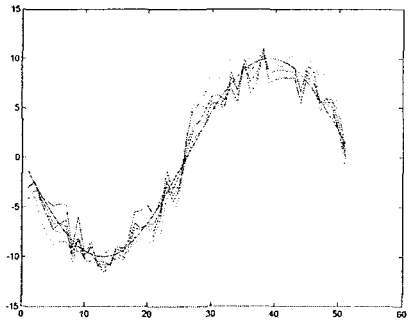


Figure A-8 Signaux théoriques et mesurés

On effectue un apprentissage sur une base de 6 points avec simulation sur l'ensemble des données. On entraîne ensuite le réseau avec la totalité des points, soit 51 points, et on simule sur l'ensemble de tous les points, soit 51 points.

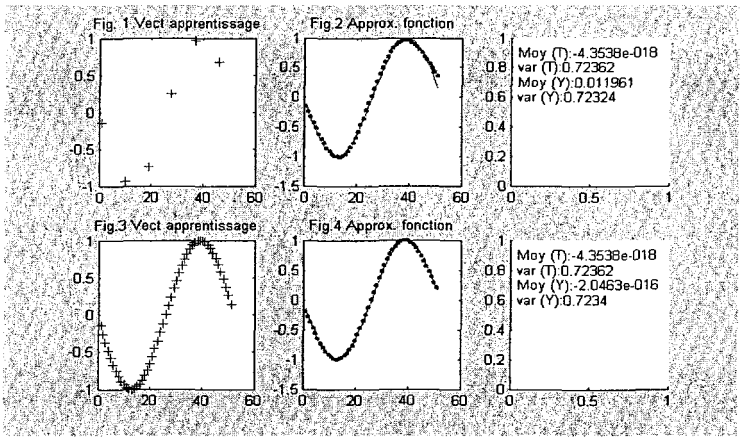


Figure A-9 Approximation de fonction (courbe théorique)

Les moyennes et variances sont indiquées pour chacune des approximations; Moy(T) et var(T) pour la courbe théorique et Moy(Y) et var(Y) pour la sortie du réseau RBF. À la lumière de ces graphiques, on remarque que le réseau RBF fourni une très bonne approximation sur la fonction lorsque le nombre de points augmente, cela étant du à un facteur de régulation dans la méthode d'apprentissage.

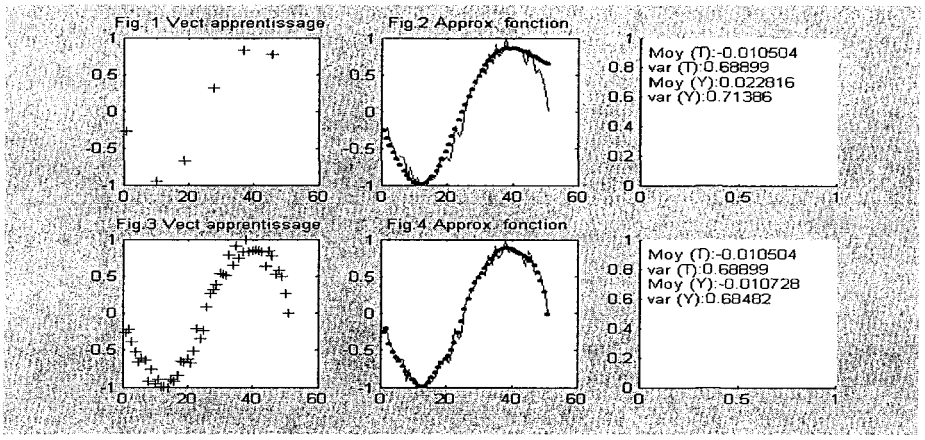


Figure A-10 Approximation de courbe(valeur mesurée)

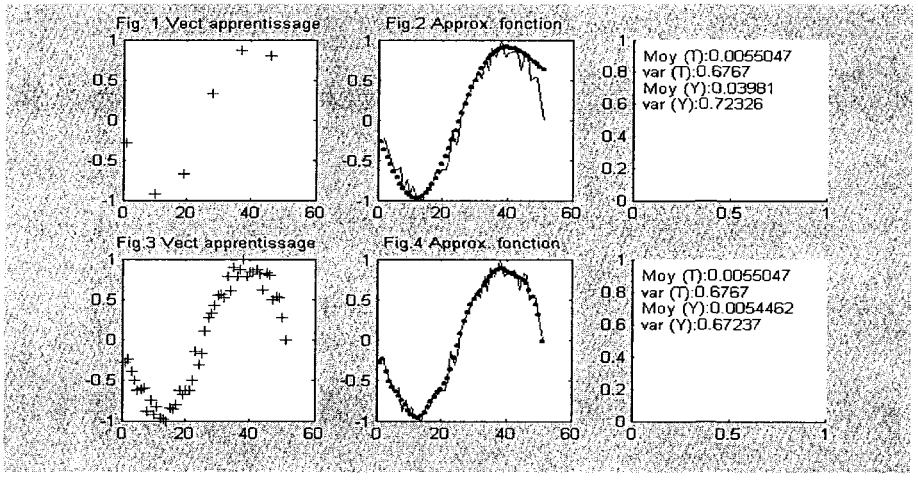


Figure A-11 Approximation de courbe (valeur bruitée b1)

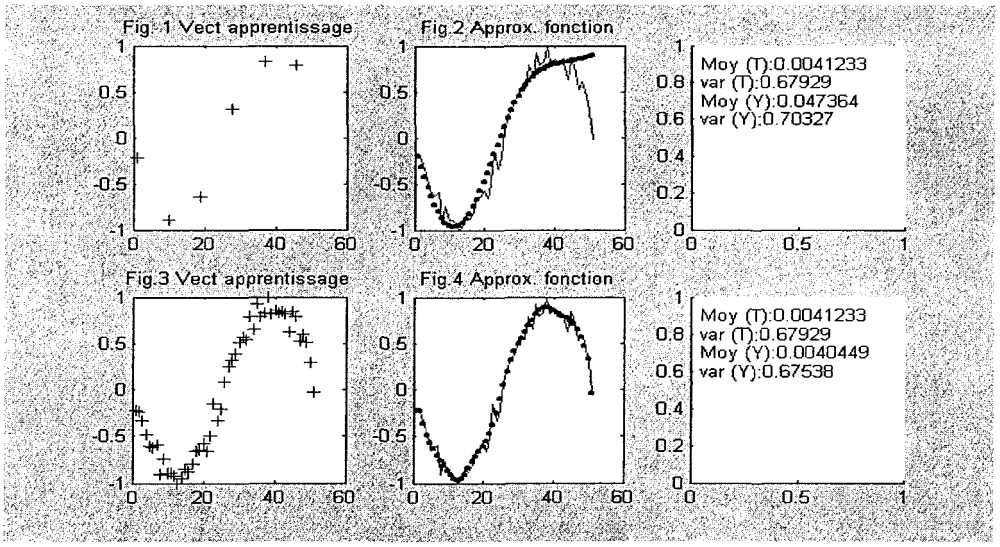


Figure A-12 Approximation de courbe (valeur bruitée b2)

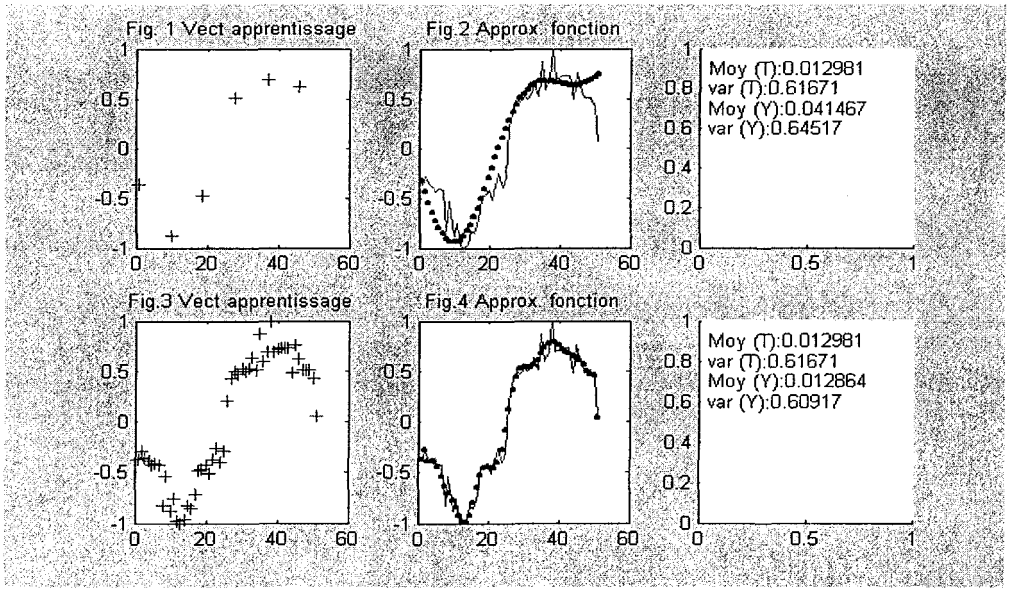


Figure A-13 Approximation de courbe (valeur bruitée b3)

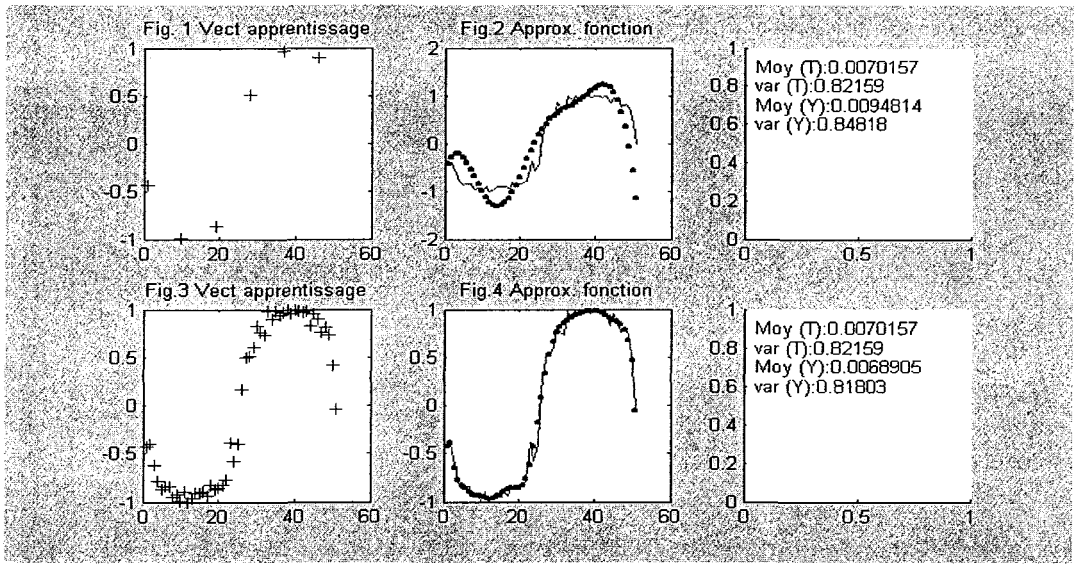


Figure A-14 Approximation de courbe (valeur bruitée b4)

Cet exemple sert uniquement comme référence pour la reconnaissance des signaux à l'aide d'un réseau RBF.

La méthode Taguchi des plans d'expériences

La méthode de Taguchi utilise le plan d'expériences comme outil statistique pour l'optimisation de processus. Cette optimisation permet un choix approprié des niveaux associés aux facteurs expérimentaux. La méthode de Taguchi permet de faire en sorte que le choix des niveaux soient moins sensible aux facteurs de bruit.

L'exemple suivant servira à faire comprendre la philosophie de base de la méthode de Taguchi. Cet exemple provient du livre « Design for quality » de Robert H. Lochner et Joseph E. Matar. À la figure A-15, on représente la relation qu'il y a entre la température d'un four et le temps de cuisson d'une pièce moulée.

Supposons que lors d'un procédé de cuisson, la tolérance sur la température soit une fonction du temps de cuisson dans le four. La relation de tolérance est représentée par la bande située entre les deux courbes de la figure A-15.

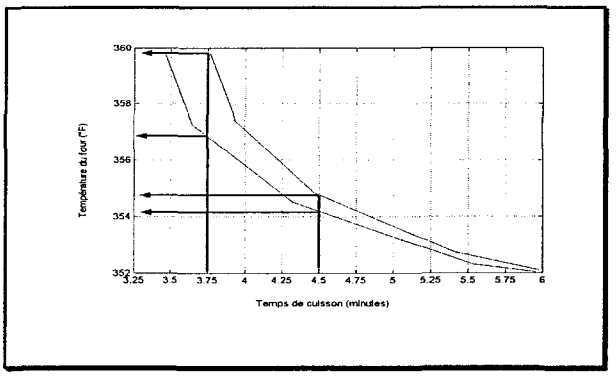


Figure A-15 Relation entre la température du four et le temps de cuisson

Par exemple, si une spécification stipule que l'on doit avoir un temps de cuisson de 4,5 minutes, alors la température doit être maintenue entre 354 et 354,8, soit un intervalle de tolérance de 0,8 °F. Il est possible que le four ne soit pas en mesure d'avoir cette tolérance. Une solution serait d'acheter un nouveau four ou un nouveau système de contrôle de la température. Une autre possibilité serait de modifier le temps de cuisson pour 3,75 minutes. De cette manière, la température devra se situer entre 357 °F et 360 °F, ce qui correspond à une tolérance acceptable. Si le four peut garder cette tolérance, la décision la plus économique sera d'ajuster la spécification. Cela fait en sorte que le procédé sera moins sensible à la variation de température du four. C'est la philosophie que nous devons adopter pour éviter des coûts supplémentaires et une meilleure gestion de la qualité.

En résumé, nous disons que la méthode de Taguchi est de rendre insensible aux bruits la valeur cible du résultat (soit la qualité d'une pièce ou d'un procédé) à partir de la connaissance d'un certain plan d'expériences. Nous allons maintenant expérimenter la méthode des plans factoriels complets à l'aide d'un exemple concret qui présentera une première approche des plans d'expérience.

Le plan standard vs plan expérimental à deux niveaux

Lors d'une expérience scientifique traditionnelle, un ingénieur expérimenté sait comment vérifier la qualité d'un produit ou d'un procédé en ne faisant varier qu'un facteur à la fois en gardant fixe les autres facteurs. Cette façon de faire n'apporte aucune solution optimale sur la performance d'un produit ou d'un procédé. Nous allons l'expliquer à partir d'un exemple simple.

Supposons que la qualité d'un procédé est une fonction de trois facteurs : A, B et C. Nous cherchons à maximiser la qualité du procédé selon deux valeurs différentes pour chacun des trois facteurs tout en minimisant les pertes. Si nous prenons l'approche traditionnelle, soit celle de faire varier un facteur à la fois tout en gardant les autres constants, nous aurons à effectuer quatre (4) expériences avec les niveaux indiqués selon la figure A-16, où les codes 1 et 2 représentent les deux niveaux à partir desquels chacun des facteurs sera initialisé durant l'expérience.

	Niveau des facteurs			
No. Expérience	A	B	C	Réponse (qualité)
1	1	1	1	Q1
2	2	1	1	Q2
3	1	2	1	Q3
4	1	1	2	Q4

Figure A-16 Plan d'expérience standard avec variation d'un seul facteur

De manière générale, le niveau 1 est appelé le niveau bas pour le facteur considéré et le niveau 2, le niveau haut du facteur considéré. Supposons maintenant que nous réalisions les quatre expériences et que nous obtenions les résultats présentés à la figure A-17.

	Niveau des facteurs			
No. Expérience	A	B	C	Réponse (qualité)
1	1	1	1	125
2	2	1	1	100
3	1	2	1	130
4	1	1	2	105

Figure A-17 Exemple de réponses selon l'état des niveaux

À partir de ces informations, on peut conclure que la qualité du procédé est maximisée, pour la valeur Q égale à 130, lorsque les facteurs A et C sont au niveau 1 et le facteur B au niveau 2. À première vue, il apparaîtrait que cette combinaison serait la seule qualifiable pour maximiser le procédé ou la qualité. Cela suggère également que la variation du facteur A du niveau 1 au niveau 2 tout en gardant les facteurs B et C au niveau 1 fait décroître la qualité du procédé par 125-105=20.

Supposons maintenant que les facteurs B et C interagissent ensembles. Cela veut dire que l'effet d'un facteur sur la qualité du procédé est affecté par le niveau de l'autre facteur.

Ce type d'expérience ne prend pas en compte cette combinaison de facteurs car si nous considérions cet effet, il faudrait choisir les niveaux suivants pour obtenir une maximisation, soit A et B au niveau 2 et C au niveau 1, ce qui est très différent du niveau des facteurs selon la première analyse. Nous pouvons dire que le plan d'expérience standard n'est pas fiable et qu'il faudrait pousser plus loin l'expérimentation. Pour éviter de tomber dans ce piège coûteux en temps et argent, nous utiliserons des méthodes statistiques appropriées à partir d'un plan d'expériences pouvant ressortir les effets reliés par la combinaison de facteurs.

Le plan expérimental à trois facteurs à deux niveaux

Dans un plan expérimental à trois facteurs et deux niveaux, on doit effectuer 2^3 expériences, soit 8 expériences pour avoir la quantité de données permettant de couvrir toutes les possibilités d'interactions pouvant affecter la réponse. Si nous avions 4 facteurs chacun associés à trois niveaux, nous aurions à effectuer 3^4 expériences, soit 81 expériences. Nous verrons plus loin qu'il existe des tables prédéfinies permettant un plus petit nombre d'expériences, les tables étant construites de manière à minimiser le bruit. Pour une expérience comptant sur la variation de trois facteurs selon deux niveaux, les expériences sont ordonnées selon la figure A-18.

Numéro de l'expérience	Niveau des facteurs			Réponse
	A	B	C	
1	1	1	1	Q1
2	1	1	2	Q2
3	1	2	1	Q3
4	1	2	2	Q4
5	2	1	1	Q5
6	2	1	2	Q6
7	2	2	1	Q7
8	2	2	2	Q8

Figure A-18 Plan d'expérience 3 facteurs à 2 niveau

Le tableau de la figure A-18 prend en considération toutes les possibilités de niveau associées à l'ensemble des facteurs pour les combinaisons expérimentales. Il est entendu que pour un plan factoriel complet, plus nous aurons de facteurs et de niveaux et plus grand sera le nombre d'expériences qu'il faudra effectuer pour obtenir une meilleure précision sur les effets, amenant par le fait même une optimisation maximale du procédé ou de la qualité.

Pour obtenir toutes les combinaisons et faire en sorte que la table de réponse soit optimale, nous devons la construire selon une méthode appropriée. Pour des facteurs à deux niveaux, on construit le tableau de la figure A-19 par référence à la logique booléenne pour obtenir l'orthogonalité des colonnes, on dit alors que le plan d'expériences est orthogonal.

Nous verrons plus loin dans cette section comment déterminer si les colonnes sont orthogonales ou non. Pour trois facteurs à niveaux binaires, nous obtenons la bonne combinaison d'expériences en se fiant à la règle suivante: nous remplaçons respectivement les valeurs 1 et 2 par 0 et 1 et nous construisons la table selon un ordre croissant décimal en utilisant la relation $S = b_2 * 2^2 + b_1 * 2^1 + b_0 * 2^0$ où b_2, b_1 et b_0 prennent les valeurs 0 ou 1.

No. de l'expérience		Niveau des facteurs		
	Valeur de la combinaison (1,2,4)	A Décimal 4	B Décimal 2	C Décimal 1
1	0=(0*4)+(0*2)+(0*1)	0	0	0
2	1=(0*4)+(0*2)+(1*1)	0	0	1
3	2=(0*4)+(1*2)+(0*1)	0	1	0
4	3=(0*4)+(1*2)+(1*1)	0	1	1
5	4=(1*4)+(0*2)+(0*1)	1	0	0
6	5=(1*4)+(0*2)+(1*1)	1	0	1
7	6=(1*4)+(1*2)+(0*1)	1	1	0
8	7=(1*4)+(1*2)+(1*1)	1	1	1

Figure A-19 Tableau orthogonal de 3 facteurs à 2 niveaux en logique booléenne

Le plan factoriel complet se construit toujours de cette manière. Nous verrons plus loin qu'il existe des plans permettant d'effectuer moins d'expériences que les plans factoriels complets. Nous devons néanmoins utiliser les statistiques appropriées pour déterminer les niveaux qui apporteront une maximisation ou une minimisation du procédé ou de la qualité.

Une fois la construction terminée, nous remplaçons les valeurs 0 et 1 par leurs valeurs de niveau de départ. Nous utiliserons le même procédé pour les tables de facteurs ayant plus de deux niveaux. Par exemple, pour une table de trois facteurs ayant 3 niveaux, soit l'ensemble de niveaux (0,1,2), les combinaisons seront déterminées par la relation

$$S = b_2 * 3^2 + b_1 * 3^1 + b_0 * 3^0$$

Le tableau de la figure A-20 donne les valeurs des combinaisons sous forme binaire.

No. de l'expérience		Niveau des facteurs		
	Valeur de la combinaison (1,3,9)	A	B	C
27 expériences		Décimal 9	Décimal 3	Décimal 1
1	0=(0*9)+(0*3)+(0*1)	0	0	0
2	1=(0*9)+(0*3)+(1*1)	0	0	1
3	2=(0*9)+(1*3)+(0*1)	0	0	2
4	3=(0*9)+(1*3)+(1*1)	0	1	0
5	4=(1*9)+(0*3)+(0*1)	0	1	1
...	...	...	...	...
26	25=(2*9)+(2*3)+(1*1)	2	2	1
27	26=(2*9)+(2*3)+(2*1)	2	2	2

Figure A-20 Table de résultat pour 3 facteurs à 3 niveaux

On retrouve des tables d'expériences configurées selon un nombre fini d'expériences. On les appelle des tables L9, L18 et L27. Ces tables sont configurées selon une minimisation du bruit par combinaison de facteurs.

Ces tables de résultat contiennent respectivement 9 expériences avec 4 facteurs à trois niveaux pour la table L9, 18 expériences avec 7 facteurs à trois niveaux plus un huitième facteur à deux niveaux pour la table L18 et finalement 27 expériences avec 13 facteurs à trois niveaux chacun pour la table L27. Les statistiques reliées à ces tables sont différentes par rapport aux statistiques des plans d'expériences standards et factoriels.

Ordre standard	Colonne			
Expérience	1	2	3	4
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Figure A-21 Plan d'expériences L9

Dans ce plan d'expériences, les nombres 1,2 et 3 représentent les trois niveaux pour chacun des quatres facteurs. Les tables L9, L18 et L27 sont configurées pour que les colonnes soient orthogonales entre elles. Nous devons faire neuf (9) manipulations pour compléter le plan d'expériences et ensuite déterminer par un calcul simple les facteurs qui influenceront sur les résultats avec une minimisation du bruit.

Principe d'orthogonalité

Brièvement, l'orthogonalité permet d'estimer la moyenne des effets des facteurs sans craindre que les résultats subissent une distorsion par l'effet des autres facteurs. Le principe d.orthogonalité s'applique uniquement pour les plans d'expériences autres que les tables L9, L18 et L27. On applique une technique simple permettant de savoir si les facteurs sont orthogonaux : Pour des facteurs à deux niveaux, on remplace les valeurs 1 et 2 des niveaux par les valeurs --1 et 1 respectivement. C'est la notation que préfèrent les statisticiens nord-américains. On multiplie les valeurs correspondantes de chacune des deux colonnes et on en fait la somme. Si la somme est égale à zéro, les colonnes sont orthogonales et les effets représentés par ces colonnes sont alors dits orthogonaux.

Par exemple, à partir du tableau de la figure A-18, nous construisons le tableau de la figure A-22 dans lequel la valeur 1 est remplacée par –1 et la valeur 2 par +1. Si nous avions trois niveaux au lieu de deux, nous aurions remplacé respectivement les valeurs 1,2 et 3 par –1,0 et 1. Pour un nombre de facteurs et de niveaux plus élevés, par exemple pour 4 niveaux, nous choisirions –2,-1,1 et 2 et par la suite trouver les bonnes combinaisons qui rendraient les colonnes orthogonales entre elles. Les tableaux L9, L18 et L27 sont déjà conçus pour être orthogonaux.

No. de l'expérience	Niveau des facteurs			Réponse
	A	B	C	
1	-1	-1	-1	Q1
2	-1	-1	1	Q2
3	-1	1	-1	Q3
4	-1	1	1	Q4
5	1	-1	-1	Q5
6	1	-1	1	Q6
7	1	1	-1	Q7
8	1	1	1	Q8

Figure A-22 Matrice d'expériences avec la notation (-1,1)

Prenons les colonnes A et B du tableau de la figure A-22 et nous vérifions si les colonnes sont orthogonales entre elles. Le calcul s'effectue comme suit :

$$\text{Ortho(A,B)} = \begin{aligned}&(-1)(-1)_1 + (-1)(-1)_2 + (-1)(1)_3 + (-1)(1)_4 + \\ &+ (1)(-1)_5 + (1)(-1)_6 + (1)(1)_7 + (1)(1)_8 = 0\end{aligned}$$

Les colonnes A et B sont effectivement orthogonales. Nous pouvons procéder de même pour les colonnes A et C ainsi que les colonnes B et C.

$$\text{Ortho(A,C)} = \begin{aligned}&(-1)(-1)_1 + (-1)(1)_2 + (-1)(-1)_3 + (-1)(1)_4 + \\ &+ (1)(-1)_5 + (1)(1)_6 + (1)(-1)_7 + (1)(1)_8 = 0\end{aligned}$$

$$\text{Ortho(B,C)} = \begin{aligned}&(-1)(-1)_1 + (-1)(1)_2 + (1)(-1)_3 + (1)(1)_4 + \\ &+ (-1)(-1)_5 + (-1)(1)_6 + (1)(-1)_7 + (1)(1)_8 = 0\end{aligned}$$

Il est important de choisir la bonne combinaison de niveau dans un tableau dont les facteurs sont à deux niveaux. Nous allons maintenant calculer l'effet d'un facteur sélectionné sur la réponse du système. L'effet d'un facteur est le changement occasionné sur la réponse lorsque le niveau du facteur varie du niveau bas au niveau haut. On estime l'effet de chacun des facteurs en calculant la moyenne de la réponse pour le facteur au niveau bas et la moyenne de la réponse pour le facteur au niveau haut. La différence entre les deux valeurs moyennes nous donnera l'effet de ce facteur sur la réponse du système.

Effet de A = (moyenne des réponses au niveau 2) - (moyenne des réponses au niveau 1)

$$\text{Effet de A} = \overline{A}_2 - \overline{A}_1$$

$$\text{Effet de A} = \frac{Q_5 + Q_6 + Q_7 + Q_8}{4} - \frac{Q_1 + Q_2 + Q_3 + Q_4}{4}$$

Les effets de chacun des facteurs sont calculés de la même manière. Calculons l'effet de B et C.

Effet de B = (moyenne des réponses au niveau 2) - (moyenne des réponses au niveau 1)

$$\text{Effet de B} = \overline{B}_2 - \overline{B}_1$$

$$\text{Effet de B} = \frac{Q_3 + Q_4 + Q_7 + Q_8}{4} - \frac{Q_1 + Q_2 + Q_5 + Q_6}{4}$$

Effet de C = (moyenne des réponses au niveau 2) - (moyenne des réponses au niveau 1)

$$\text{Effet de C} = \overline{C}_2 - \overline{C}_1$$

$$\text{Effet de C} = \frac{Q_2 + Q_4 + Q_6 + Q_8}{4} - \frac{Q_1 + Q_3 + Q_5 + Q_7}{4}$$

Le tableau de la figure A-23 illustre la démarche globale pour obtenir les effets de chacun des facteurs sur la réponse du système.

L'effet de chacun des facteurs ayant été déterminé, nous traçons une représentation graphique de l'effet de ces facteurs. À partir de ce graphique, nous calculons les valeurs maximales et minimales de la réponse correspondant aux effets de chacun des facteurs.

No. de l'expérience	Valeur de la réponse observée	Facteurs					
		A		B		C	
		1	2	1	2	1	2
1	Q1	Q1		Q1		Q1	
2	Q2	Q2		Q2			Q2
3	Q3	Q3			Q3	Q3	
4	Q4	Q4			Q4		Q4
5	Q5		Q5	Q5		Q5	
6	Q6		Q6	Q6			Q6
7	Q7		Q7		Q7	Q7	
8	Q8		Q8		Q8		Q8
Total							
Nombre de valeurs	8	4	4	4	4	4	4
Moyenne	Ym	A1	A2	B1	B2	C1	C2
Effet		A2-A1		B2-B1		C2-C1	

Figure A-23 Table des effets

Nous pouvons visualiser les effets des facteurs à l'aide d'un graphique utilisé par Kackar et Shoemaker. La représentation graphique montre les écarts entre les niveaux par rapport à la valeur moyenne des 8 réponses obtenues. Dénotons par F1 et F2 la moyenne des niveaux 1 et 2 pour le facteur A, S1 et S2 pour le facteur B, de même que T1 et T2 pour le facteur C.

La figure A-31 est un exemple de représentation graphique de ces effets.

Nous pouvons calculer la réponse maximale en additionnant les différences des effets positifs avec la moyenne et en ajoutant à la valeur moyenne, même chose pour la réponse minimale.

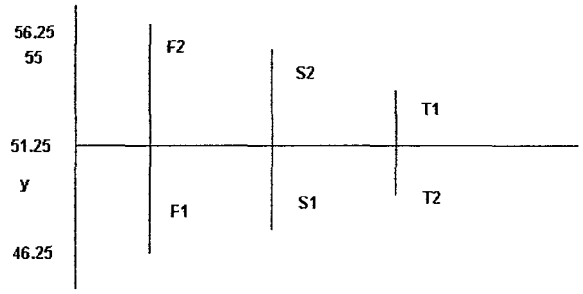


Figure A-31 Graphe des effets de chacun des facteurs

Les calculs suivants permettent de déterminer la réponse maximale et minimale,

$$Q_{\max} = Q_{moy} + (F_2 - Q_{moy}) + (S_2 - Q_{moy}) + (T_1 - Q_{moy})$$
$$Q_{\max} = 51.25 + (56.25 - 51.25) + (55 - 51.25) + (53.75 - 51.25)$$
$$Q_{\max} = 62.5$$

$$Q_{\min} = Q_{moy} + (F_1 - Q_{moy}) + (S_1 - Q_{moy}) + (T_2 - Q_{moy})$$
$$Q_{\min} = 51.25 + (46.25 - 51.25) + (47.5 - 51.25) + (48.75 - 51.25)$$
$$Q_{\min} = 40$$

Si nous voulons maximiser, nous devons prendre comme valeurs les facteurs F2,S2 et T1 tandis que pour minimiser, nous devons prendre les valeurs des facteurs F1,S1,T2. Lors de la réplication des plans d'expériences, la moyenne de la somme des réponses doit approcher la moyenne générale de chacune des répliques.

Au lien de diviser la somme des réponses par 8 on la divise par 16 si nous avons deux répliques et par 24 si nous avons trois répliques. De même, au niveau des effets, on calcul la moyenne en prenant la somme des réponses et on divise par 8 pour deux répliques ou par 12 pour 3 répliques.

Interaction entre les facteurs

Pour expliquer l'interaction des facteurs, nous utiliserons le tableau de la figure A-23. Dans cet exemple, les trois facteurs ont été traités de manière additive. Cela veut dire que chaque facteur qui contribue à la variation de la réponse a subit une variation indépendamment de chacun des deux autres. Ce qu'il incombe de savoir ici, c'est comment faire pour déterminer l'interaction de deux ou trois facteurs dans un plan d'expérience sans pour autant ajouter de nouvelles expériences. Lorsque l'effet d'un facteur est influencé par le niveau d'un autre facteur, nous disons qu'il y a un effet d'interaction entre les deux facteurs.

On peut voir comment évolue un facteur par rapport à un autre en visualisant un graphe d'influence comme sur la figure A-32. C'est de cette manière que nous pouvons comprendre comment déterminer les interactions entre facteurs.

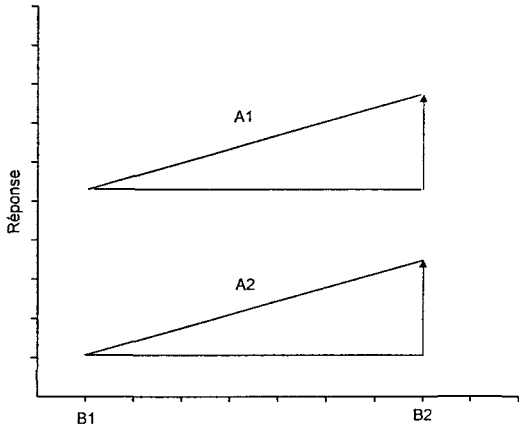


Figure A-32 Graphe d'influence ou d'interaction

Ce graphe se lit ainsi : B1 représente le niveau 1 du facteur B, B2 le niveau 2 du facteur B tandis que A1 est le niveau un du facteur A, On trace alors un premier point correspondant à la réponse du système donnée par les niveaux B1-A1.

On trace ensuite un second point correspondant à la réponse du système pour B2-A1 et on joint par la suite les deux points pour former une droite A1-B. Nous faisons de même pour la droite de l'effet A2-B

Par convention, les effets d'interactions entre A et B sont obtenues par la moitié de la différence entre les deux effets :

$$\textbf{Interaction AB} = \frac{\textbf{ΔrépA2B - ΔrépA1B}}{\textbf{2}}$$

Par exemple, pour le graphe de la figure 4.30, nous avons delta(rép)A2B=6 et delta(rép)A1B=6 d'où un effet d'interaction qui est égal à :

$$\textbf{Interaction AB} = \frac{\textbf{6 - 6}}{\textbf{2}} = \textbf{0}$$

Il n'y a donc aucune interaction entre les facteurs A et B. Dans d'autres cas, on pourra avoir une interaction positive ou négative.