50376
1992
17

N° d'ordre 868

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

pour l'obtention du titre de

DOCTEUR

En Productique: Automatique et Informatique Industrielle

par

Xianyi ZENG

PILOTAGE DES PROCESSUS DYNAMIQUES PAR ECHANTILLONNAGE STATISTIQUE

Soutenue le 4 février devant la Commission d'Examen:

MM.	P. VIDAL	Président
	B. DUBUISSON	Rapporteur
	J.-G. POSTAIRE	Rapporteur
	C. VASSEUR	Directeur de Recherche
Mme	M.-C. VIANO	Examinateur

AVANT-PROPOS

Le travail exposé dans ce mémoire a été réalisé au centre d'Automatique de l'Université de Lille I sous la direction de Monsieur le Professeur Christian VASSEUR. Avant de présenter cette étude, je tiens à exprimer ma reconnaissance à toutes les personnes qui m'ont aidé dans ce travail.

J'exprime ma gratitude à Monsieur le Professeur Pierre VIDAL, qui m'a accueilli dans son laboratoire et je le remercie d'avoir accepté la présidence de mon jury de thèse.

J'adresse mes sincères remerciements à Monsieur Christian VASSEUR, Professeur à l'Université de Lille I et Directeur de l'Ecole Nationale Supérieure des Arts et Industries Textiles de Roubaix, qui m'a constamment aidé, guidé, conseillé tout au long de l'élaboration de cette étude.

J'adresse mes plus vifs remerciements à Monsieur le Professeur Bernard DUBUISSON, Directeur du Centre d'Automatique et Traitement du Signal de l'U.T. de Compiègne, pour l'intérêt qu'il a bien voulu porter au jugement de ce travail en acceptant d'être l'un de mes rapporteurs.

Je remercie également Monsieur Jack-Gérard POSTAIRE, Professeur à l'Université de Lille I, pour avoir accepté de juger le contenu de ce mémoire et pour sa présence parmi les membres de jury en tant que rapporteur.

J'adresse de même toute ma reconnaissance à Madame Marie-Claude VIANO, Professeur à l'Université de Lille I, pour l'intérêt qu'elle a témoigné à ces travaux en tant que membre du jury.

L'ensemble de mes travaux a pu être mené à bien grâce à l'aide de tout le personnel du Centre d'Automatique de Lille I, auquel je tiens à adresser ma gratitude.

TABLE DES MATIERES

INTRODUCTION GENERALE

Ce mémoire a pour cadre l'étude des processus dynamiques basée sur l'observation d'échantillons statistiques prélevés sur ces processus.

Dans la pratique, le fonctionnement d'un processus est souvent caractérisé par un ensemble de performances externes évoluant dans le temps. Ces performances observées fournissent certaines informations permettant d'analyser la structure statistique interne du processus. Les outils utilisés pour cela sont, par exemple, la classification temps réel, l'estimation des paramètres du processus, le filtrage des données observées, etc. Ces informations permettent également de modifier la structure interne du processus afin d'obtenir une sortie imposée sans établir de modèle mathématique.

Dans les opérations présentées dans ce mémoire, nous avons développé la possibilité d'intégrer d'une manière dynamique les nouvelles informations fournies par l'observation du processus afin d'enrichir une base d'apprentissage pré-existante. Cette base d'apprentissage est parfois incomplète et ne contient pas tous les modes de fonctionnement du système étudié. Au cours de l'évolution du processus dynamique, les informations intégrées dans cette base permettent donc à celle-ci de refléter de plus en plus fidèlement les caractéristiques du processus.

Le mémoire est divisé en deux parties intitulées respectivement:

- Analyse des processus dynamiques par observation d'échantillons statistiques

- Pilotage des processus par observation d'échantillons statistiques

La première partie analyse le fonctionnement du processus (Chapitre I) et développe une approche pour l'estimation des paramètres statistiques caractérisant les modes de fonctionnement de ce processus (Chapitre II). Ensuite, afin de prendre en compte les bruits inhérents à toute prise d'information, on propose dans le Chapitre III une méthode

d'estimation des paramètres à partir d'échantillons fortement perturbés. L'estimation de ces paramètres permet alors de développer des critères de fusion et de scission des modes de fonctionnement au cours de l'évolution du processus (Chapitre IV).

Dans la deuxième partie, deux algorithmes de pilotage fonctionnant sur le principe de l'auto-organisation par apprentissage sont proposés. Le premier algorithme utilise un automate stochastique (Chapitre V) et le deuxième s'organise autour de réseaux neuronaux artificiels (Chapitre VI). Enfin une étude de comparaison des deux approches est présentée dans le conclusion de cette partie.

PARTIE I

ANALYSE DES PROCESSUS DYNAMIQUES PAR OBSERVATION D'ECHANTILLONS STATISTIQUES

INTRODUCTION

Le fonctionnement d'un processus dynamique peut se traduire par un ensemble de manifestations externes évoluant dans le temps telles que bruits, déplacements, tensions électriques, pressions, températures, etc [VASSEUR, 1982].

Dans un processus dynamique, on peut distinguer les changements continus traduisant des tendances d'évolution de caractéristiques continues et les changements discontinus, traduisant soit des ruptures d'évolution de caractéristiques continues (événements), soit des modifications de position de caractéristiques discontinues (transitions) [WALLISER, 1977]. Les processus étudiés dans ce mémoire sont de type discontinu dans le temps. Tout phénomène temporel peut être considéré comme la succession dans le temps d'événements caractéristiques. Chaque événement est alors caractérisé par un certain nombre de paramètres et de mesures prises sur ces paramètres.

Les manifestations externes de ces événements permettent à l'observateur d'entrer en communication avec le processus. L'observation du processus dynamique s'effectue selon des périodes d'échantillonnage T_e et chaque événement temporel est caractérisé par les mesures des paramètres associées à cette période.

Dans cette partie, nous décrivons le fonctionnement de la procédure d'analyse du système par échantillonnage, l'estimation des paramètres caractéristiques, la fusion et la scission des classes d'échantillons. Cette procédure concerne non seulement la perception des messages émis par le processus dynamique étudié, mais aussi l'interprétation et la compréhension de ces messages. L'objectif final est de décider de l'appartenance des échantillons à telle ou telle classe caractéristique de tel ou tel mode de fonctionnement.

Le schéma bloc de la fig-1.1 résume le déroulement des différentes phases à effectuer pour analyser le processus en utilisant uniquement les données (échantillons statistiques) prélevées sur le système.

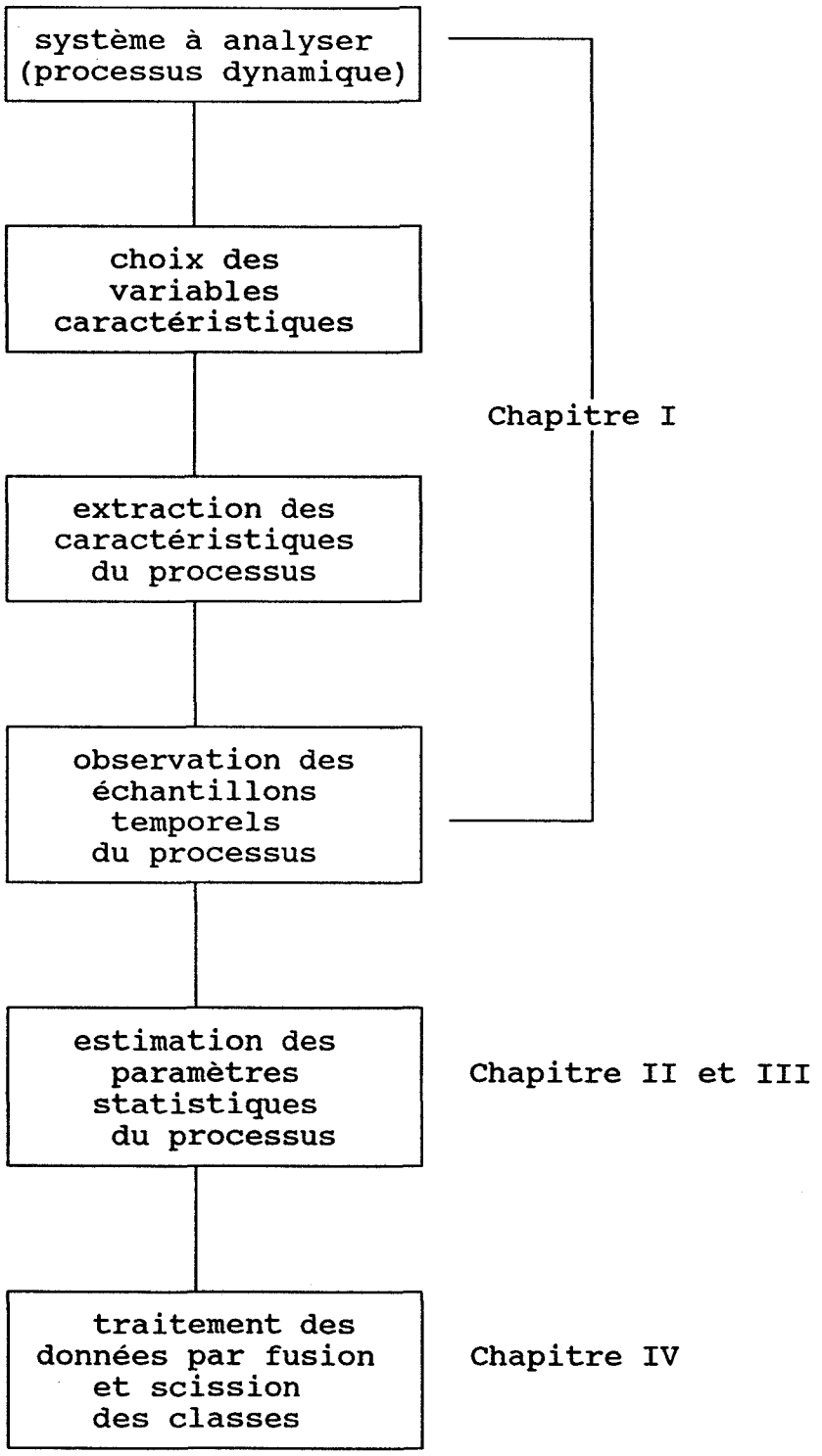


fig-1.1 schéma bloc de l'analyse du processus

Les chapitres suivants présentent les fonctions apparaissant dans la fig-1.1.

CHAPITRE I

MODELISATION ET ANALYSE DES SYSTEMES ETUDIES

II.1 ARCHITECTURE DES PROCESSUS DYNAMIQUES

II.1.1 MODELE ADOPTE

Le processus étudié est considéré comme un système indépendant qui procéde à des échanges avec son environnement. Dans ce sens, le système est influencé par un ensemble d'entrées. Par ailleurs, il exerce également une influence sur son environnement par l'intermédiaire d'un ensemble de sorties.

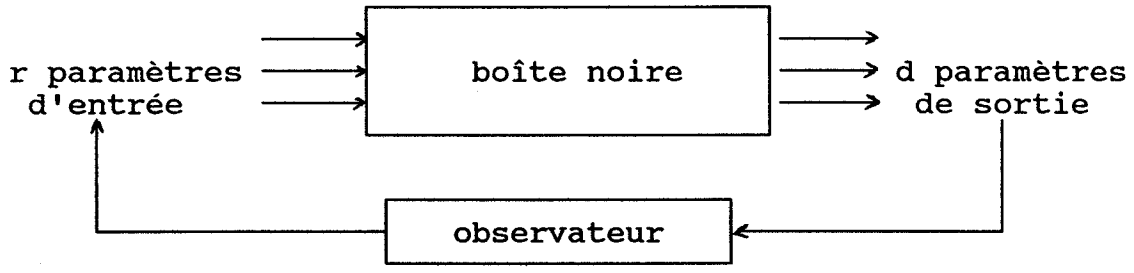


fig-1.2 modèle du processus et observation

On ne considère donc que le passage observé entre entrées et sorties sans s'intéresser aux mécanismes internes qui réalisent cette transition. Ce système est donc du type "boîte noire". [WALLISER, 1977].

II.1.2 BOITE NOIRE

Le domaine d'application du modèle "boîte noire" est très vaste. Un premier exemple pratique nous est fourni par l'étude d'un ensemble complexe de relations entre des essais et des résultats obtenus sur un groupe de machines en fonctionnement qu'on ne peut pas démonter. Un deuxième exemple est celui du

psychologue qui étudie les réactions d'un rat dans un labyrinthe en agissant sur l'animal au moyen de stimuli variés, et en observant les comportements qui en découlent. En analysant les observations, il peut améliorer la compréhension des mécanismes nerveux. [ASHBY, 1958], [BERTALANFFY, 1973].

La fig-1.2 montre comment un observateur peut se coupler au processus dynamique, en agissant sur la boîte noire et en observant les effets de stimuli. Dans ces conditions, le couple observateur - boîte noire constitue un système bouclé complet. L'analyse ou la commande du processus peuvent alors être basées sur l'observation des sorties.

Pour que ce couplage soit réalisé de manière bien définie, il faut préciser les entrées et les sorties de la boîte noire.

II.1.3 VARIABLES D'ENTREE ET VARIABLES DE SORTIE

On fait l'hypothèse que la relation entrée-sortie au sein de la boîte noire est stationnaire. Le système réalise une transformation constante entre le flux d'entrée et le flux de sortie. De plus, les espaces d'entrée et de sortie sont considérés comme étant de dimensions finies.

Les paramètres de sortie peuvent être interprétés comme les variables caractérisant la nature du système. Ils sont contrôlés par les variables caractéristiques des paramètres d'entrée. Par conséquent, la relation entre les variables d'entrée et les variables de sortie peut être considérée comme un phénomène "action-effet". Toutes les caractéristiques obtenues à travers les observations des sorties sont les effets des paramètres d'entrée du système.

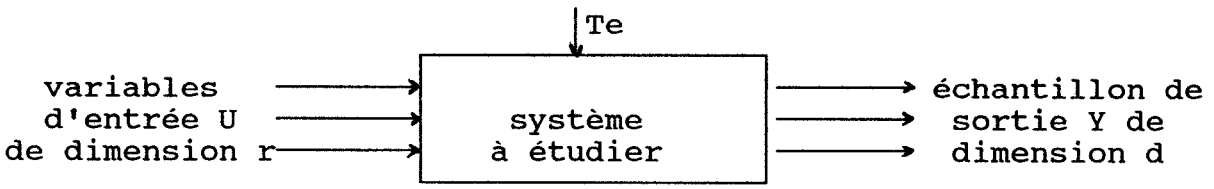
II.1.4 ECHANTILLONNAGE DES PARAMETRES CARACTERISTIQUES

En pratique, les valeurs exactes des variables de sortie ne peuvent pas être mesurées directement. Les valeurs des paramètres de sortie sont alors considérées comme des variables aléatoires qui peuvent être estimées à partir des échantillons d'observation de la sortie.

En effet, les observations sont des réalisations de ces variables aléatoires et sont obtenues par un procédé de tirage à la sortie du système. [FOURGEAUD, 1972], [FERGUSON, 1967].

Dans l'étude, on ne différentie pas les termes "échantillon" et "réalisation des échantillons". Un échantillon peut être considéré comme une observation aléatoire sur les paramètres caractéristiques du système étudié.

A chaque instant d'échantillonnage (période T_e), le système fournit un échantillon, c'est à dire, un vecteur de dimension d dans le domaine réel. L'estimation des paramètres du système se fait dans l'espace de dimension d à partir de l'ensemble de ces échantillons.



où $Y=(y_1,y_2,\ldots,y_d)^T$ et $U=(u_1,u_2,\ldots,u_r)^T$ sont deux vecteurs

fig-1.3 déroulement du processus dynamique

I.1.5 PRISE EN COMPTE DE L'EVOLUTION TEMPORELLE DU PROCESSUS

Les échantillons peuvent être considérés comme une succession dans le temps d'événements caractéristiques. Cette succession peut se diriger à partir d'une classe initiale vers une autre classe. Les paramètres dynamiques estimés par la succession des échantillons peuvent concerner plusieurs classes.

Dans la présente étude, on crée des fenêtres (sous ensembles) d'observation des échantillons de sortie du processus afin d'analyser les caractéristiques dynamiques au cours de l'évolution temporelle. Tout se passe en fait comme si la base d'apprentissage, modélisée par une fenêtre d'observation de largeur constante n, glisse échantillon après échantillon ou séquence après séquence sur l'axe temporel, au cours de l'évolution.

La méthode du glissement de fenêtre possède les avantages suivants:

- fournir une image instantanée de la situation actuelle en ne considérant que les n derniers échantillons intégrés.

- fournir un poids relatif constant à chaque échantillon.

- rendre compte des variations des paramètres caractéristiques au cours de l'évolution d'une succession d'échantillons.

- fournir un filtre pour les échantillons observés en éliminant la perturbation due à l'observation du système.

Dans chaque fenêtre, nous faisons l'hypothèse que les lois de distribution des échantillons suivent des modèles statistiques analytiques [DUDA et HART, 1973], [POSTAIRE, 1987]. Pour simplifier, nous nous limiterons à des distributions normales qui sont extrêmement courantes dans la pratique.

Sous l'hypothèse normale, la fontion de densité de sortie Y peut alors s'écrire sous la forme suivante:

$$f(Y) = \frac{1}{(2\pi)^{d/2}|S|^{1/2}}\exp[-\frac{1}{2}(Y-M)^T S^{-1}(Y-M)] \qquad (1.1.1)$$

La fonction f(Y) est définie par deux ensembles de paramètres caractéristiques du système, notés M et S:

- $M=E(Y)$ est le vecteur moyenne de la distribution de dimension d.

- $S=E((Y-M)(Y-M)^T)$ est la matrice de covariance d×d de la distribution. Evidemment, S est une matrice définie positive.

Pour simplifier, on note $f(Y)=N(M,S)$ (1.1.2)

Sous cette hypothèse, chaque fenêtre d'observation correspond à une distribution normale (nuage gaussien). Ainsi, l'évolution temporelle du processus se traduit par le glissement de la fenêtre et par l'évolution du nuage gaussien le long de l'axe temporel.

Dans le système étudié, nous avons fait l'hypothèse que les variables d'entrée influent sur les paramètres M et S. Dès

lors, au cours de l'observation temporelle, la distribution des échantillons est commandable à partir des variables d'entrée évoluant dans le temps.

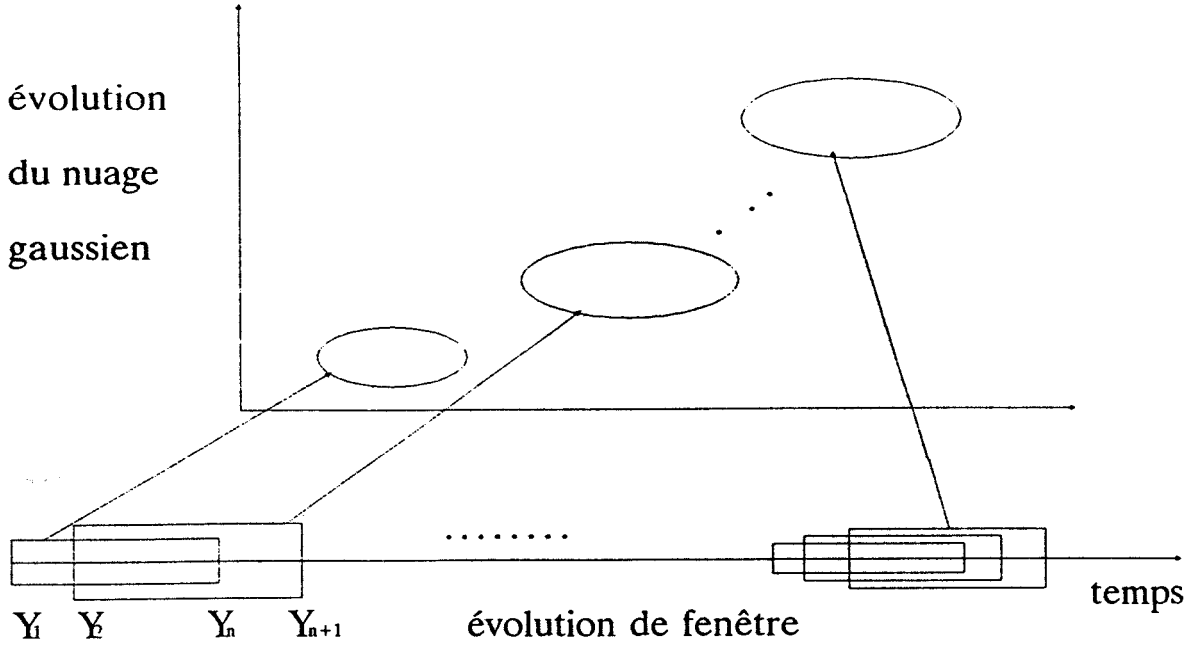


fig-1.4 glissement de fenêtre

II.1.6 OBSERVATION DES ECHANTILLONS POUR DES CLASSES DIFFERENTES

La dimension d des échantillons de sortie du système correspond au nombre d'attributs caractéristiques extraits du processus dynamique. Chaque observation fournit alors un point dans l'espace des attributs (fig-1.4) et la succession dans le temps des observations génère un nuage de points multi-modal faisant apparaître différentes classes ou modes.

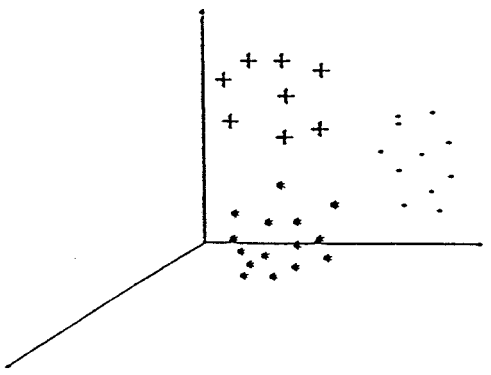


fig-1.5 observations des échantillons de sortie pour des classes différentes

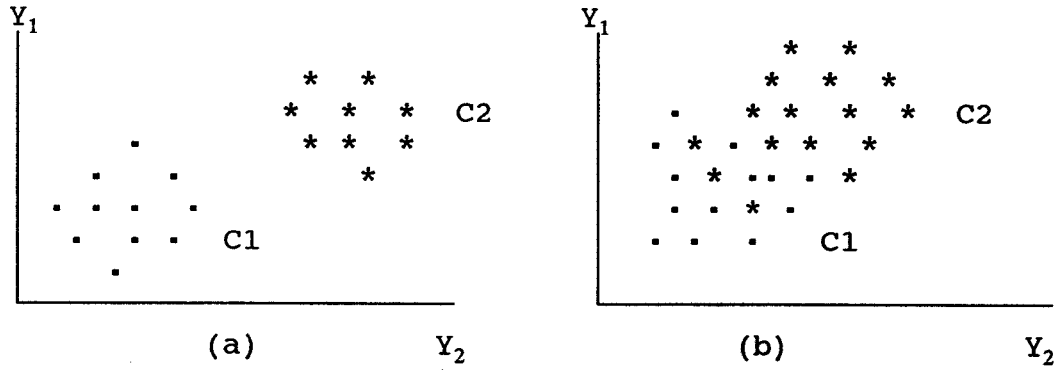
II.2 EXTRACTION DES CARACTERISTIQUES DU PROCESSUS

Dans un processus dynamique, la sélection des paramètres caractéristiques vise à mettre en évidence, à chaque instant, les propriétés principales de ce processus. Cette étape est déterminante pour la réalisation d'une bonne observation.

La sélection des meilleurs paramètres, qui sort du cadre de cette étude, doit satisfaire deux critères fondamentaux:

- améliorer la séparabilité des classes de fonctionnement afin d'obtenir un taux d'erreur de séparation aussi faible que possible. La fig-1.6 illustre les effets d'un bon choix (a) et d'un mauvais choix (b).

- minimiser le nombre de paramètres nécessaires à une bonne observation dans le but de réduire la dimension de l'espace et donc le coût de calcul.



· échantillons de la classe C1
* échantillons de la classe C2

fig-1.6 séparation de deux classes dans l'espace d'observation

(a) sélection de paramètres convenables
(b) sélection de paramètres non convenables

De nombreuses approches se sont consacrées à l'extraction des caractéristiques [FUKUNAGA et SHORT, 1980], [FUKUNAGA et SHORT, 1978], [FUKUNAGA, 1972].

I.3 OBSERVATION DES ECHANTILLONS PAR FENETRAGE

I.3.1 NOTION DE FENETRE D'OBSERVATION

La caractérisation des échantillons du processus dans le temps s'appuie sur la réalisation de fenêtres d'observation variables. A partir de ces fenêtres, on peut analyser les tendances et la dynamique des échantillons observés.

L'utilisation de fenêtres d'observation conduit à un effet de filtrage comparable au phénomène de moyenne glissante. Il est possible de définir avec précision les caractéristiques du filtrage en fonction de la fréquence d'échantillonnage et de la largeur de la fenêtre utilisée [EBERHARD, 1973].

La méthode du glissement de fenêtre est appliquée dans des domaines variés. Par exemple, afin d'analyser l'électroencéphalogramme, on peut définir une fenêtre d'observation glissant le long du signal et à l'intérieur de laquelle sont évalués les spectres successifs de l'électroencéphalogramme. Cette démarche associe à chaque fenêtre d'observation un événement "calcul du spectre" [BODENSTEIN et PRAETORIUS, 1977].

Dans [CHEBEL, 1983], la surveillance du processus se ramène à l'observation d'une suite de séquences pouvant traverser un nombre k de situations données. L'étude de l'évolution du processus est envisagée par le biais des lois de probabilites calculés sur une fenêtre glissante dans le temps comprenant un nombre fixe de séquences. Ce glissement effectué peut être d'une ou plusieurs séquences.

Dans [VASSEUR, 1982], la caractérisation des chaines chronologiques d'événements est basée sur le glissement de fenêtres variables. La mise en oeuvre de fenêtres d'observation imbriquées permet d'exploiter deux aspects utiles à la description des phénomènes temporels. Dans un premier temps, l'exploitation d'une fenêtre de largeur N se traduit par un processus de filtrage qui permet de décrire les tendances moyennes du phénomène observé. Dans un second temps, l'utilisation de fenêtres imbriquées de largeur $n \leqslant N$ permet d'évaluer la dynamique instantanée qui accompagne toute tendance moyenne.

I.3.2 ESTIMATION DES PARAMETRES A PARTIR D'UNE FENETRE

La fenêtre définie dans la Section I.1.5 est utilisée dans ce mémoire. Les paramètres statistiques peuvent être estimés à partir de cette fenêtre.

Supposons que les échantillons de sortie du système proviennent d'une même classe. On effectue l'estimation des paramètres dans (1.1.1) de la manière suivante:

En prenant une fenêtre d'échantillons $\{Y_1, Y_2, \ldots, Y_n\}$, on se propose d'utiliser ces observations pour déterminer les paramètres de la distribution N(M,S), supposée normale.

La procédure d'estimation du vecteur moyenne M et de la matrice de covariance S dans le cas multivariable est énoncée dans [ANDERSON,T.W., 1958].

On obtient finalement le résultat bien connu:

$$\begin{cases} \hat{M} = \dfrac{1}{n} \displaystyle\sum_{i=1}^{n} Y_i \\ \\ \hat{S} = \dfrac{1}{n-1} \displaystyle\sum_{i=1}^{n} (Y_i - \hat{M})(Y_i - \hat{M})^T \end{cases} \qquad (1.1.3)$$

La méthode d'estimation non biaisée est utilisée dans (1.1.3). C'est une meilleure estimation pour les paramètres M et S. Nous pouvons déduire $E(\hat{M})=M$ et $E(\hat{S})=S$ à partir des équations ci-dessus.

Pour simplifier la notation, on ne différentie pas $\hat{M}$, $\hat{S}$ (valeurs estimées) et M, S (valeurs réelles) dans l'analyse ci-dessous.

Dans le chapitre suivant, un algorithme rapide d'estimation des paramètres caractéristiques est développé en utilisant le glissement de fenêtre dans un mélange de classes différentes. Les caractéristiques dynamiques au cours de l'évolution temporelle du processus peuvent être obtenues par cet algorithme.

I.4 METHODES D'ESTIMATION DES PARAMETRES STATISTIQUES

I.4.1 METHODES D'ESTIMATION DES PARAMETRES STATISTIQUES

Dans de nombreux cas d'estimation, la distribution des échantillons est modélisée comme un mélange de classes gaussiennes. Chaque classe peut être tratée séparément et la structure du mélange peut être aisément déterminée. On peut estimer les paramètres de chaque classe en utilisant des échantillons dont les appartenances sont inconnues.

Dans [COOPER et COOPER, 1964], on montre qu'il est possible de réaliser une estimation des paramètres en établissant une adapdation effective sans supervision. Dans ce cas, les appartenances correctes des échantillons sont inconnues.

Dans [YOUNG et CORALUPPI, 1970], un algorithme d'approximation stochastique est développé pour estimer un mélange de fonctions de densité normales avec moyennes et variances inconnues.

Dans [KAZAKOS, 1977], un autre algorithme d'estimation récursif des probabilités a priori est développé en utilisant les techniques d'approximation stochastique. La convergence de cet algorithme est très rapide.

Dans les cas de données multi-dimensionnelles, on peut souvent utiliser la technique du maximum de vraisemblance. Cette méthode est précise mais elle consomme beaucoup de temps [DUDA et HART, 1973]. Dans [POSTAIRE et VASSEUR, 1981], les paramètres modaux sont estimés en déterminant des domaines concaves de distribution des échantillons. Une autre approche d'estimation est d'utiliser les équations des moments pour résoudre tous les paramètres d'un mélange gaussien sans information a priori [FUKUNAGA et THOMAS, 1983].

I.4.2 APPROCHE RECURSIVE D'ESTIMATION DES PARAMETRES

Une approche récursive dans l'estimation des paramètres permet d'intégrer d'une manière dynamique les nouvelles observations qui viennent enrichir la base d'apprentissage. Elle permet également d'affiner les estimations des paramètres statistiques fournies par les méthodes de classification auto-

matique [GU et DUBUISSON, 1985].

Les valeurs initiales des paramètres sont estimées à partir d'une base d'apprentissage (ensemble initial des échantillons d'observation) selon une méthode générale.

A chaque période de l'évolution du processus, un nouvel échantillon est intégré à la base d'apprentissage. La nouvelle estimation des paramètres s'effectue à partir des dernières valeurs de ces paramètres et de la nouvelle observation.

Le chapitre suivant est entièrement consacré à cette approche récursive. On proposera un algorithme rapide pour l'estimation des valeurs propres et vecteurs propres de la matrice de covariance de distribution des échantillons. Cet algorithme permet d'obtenir une image précise et instantanée sur l'évolution des éléments propres dans le temps.

CHAPITRE II

ESTIMATION DES PARAMETRES STATISTIQUES DU PROCESSUS

II.1 ESTIMATION RECURSIVE DES ELEMENTS PROPRES

II.1.1 INTEGRATION D'UN NOUVEL ECHANTILLON

L'estimation des paramètres statistiques de M et S peut s'effectuer à partir de (1.1.3). On calcule les éléments propres de S définis comme suit:

$$[\Lambda] = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_d \end{bmatrix}$$: matrice diagonale des valeurs propres de S

$[U] = [U_1, U_2, \ldots, U_d]$: matrice des vecteurs propres de S

S peut s'écrire sous la forme: $S = [U]\ [\Lambda]\ [U]^T$

Lorsqu'on intégre un nouvel échantillon Y_{n+1}, les nouveaux paramètres peuvent être décrits de la manière suivante:

$$M^+ = M + \frac{1}{n+1}\ (Y_{n+1} - M) \qquad (1.2.1)$$

$$S^+ = [U^+]\ [\Lambda^+]\ [U^+]^T \qquad (1.2.2)$$

où S^+ est la nouvelle matrice de covariance
$[U^+]$, $[\Lambda^+]$ sont les nouveaux éléments propres de S^+

Le passage des anciens vecteurs propres aux nouveaux vecteurs propres peut s'écrire sous la forme matricielle:

$$[U^+] = [U]\ [O] \qquad (1.2.3)$$

Evidemment, [O] est une matrice de rotation. On obtient les propriétés suivantes:

$$[O]^{-1} = [O]^T \qquad (1.2.4)$$

$$\sum_{j=1}^{d} o_{ij}^2 = \sum_{i=1}^{d} o_{ij}^2 = 1 \quad \text{pour } \forall i,j \in \{1,\ldots,d\} \qquad (1.2.5)$$

$$\sum_{j=1}^{d} o_{ij}\, o_{kj} = 0 \quad \text{pour } \forall i,k \in \{1,\ldots,d\} \text{ et } i \neq k \qquad (1.2.6)$$

Après intégration du nouvel échantillon Y_{n+1} dans l'ensemble des échantillons $\{Y_1, Y_2, \ldots, Y_n\}$, la nouvelle estimation S^+ est donnée par:

$$S^+ = \frac{1}{n} \sum_{i=1}^{n+1} (Y_i - M^+)(Y_i - M^+)^T \qquad (1.2.7)$$

En reportant (1.2.1) dans (1.2.7), il vient:

$$S^+ = \frac{n-1}{n} S + \frac{1}{n+1} (Y_{n+1} - M)(Y_{n+1} - M)^T \qquad (1.2.8)$$

II.1.2 ESTIMATION DES NOUVELLES VALEURS PROPRES

En notant $\alpha = \frac{n-1}{n}$, $\beta = \frac{1}{n+1}$, $R = \Delta Y \cdot \Delta Y^T$ avec $\Delta Y = (Y_{n+1} - M)$,

on obtient: $[U^+]\,[\Lambda^+]\,[U^+]^T = \alpha \cdot S + \beta \cdot R \qquad (1.2.9)$

L'expression ci-dessus peut s'écrire dans la base [O] et en utilisant (1.2.3) et (1.2.4):

$$[U]^T (\alpha \cdot S + \beta . R)\,[U] = [U]^T [U^+][\Lambda^+][U^+]^T [U] = [O][\Lambda^+][O]^T \qquad (1.2.10)$$

A partir de (1.2.10), en notant $\Delta Y^W = [U]^T \Delta Y = [U]^T (Y_{n+1} - M)$ et $R^W = (Y_{n+1} - M)^W (Y_{n+1} - M)^{WT}$

on a: $\alpha \cdot [\Lambda] + \beta \cdot R^W = [O]\,[\Lambda^+]\,[O]^T$

soit: $(\alpha \cdot [\Lambda] + \beta \cdot R^W)[O] = [O]\,[\Lambda^+] \qquad (1.2.11)$

En extrayant la $i^{ème}$ colonne de l'égalité matricielle (1.2.11), il vient:

$$\begin{cases} \alpha\lambda_1 \cdot o_{1i} + \beta \cdot \Delta Y_1^W (o_{1i} \cdot \Delta Y_1^W + o_{2i} \cdot \Delta Y_2^W + \ldots\ldots + o_{di} \cdot \Delta Y_d^W) = o_{1i} \cdot \lambda_i^+ \\ \alpha\lambda_2 \cdot o_{2i} + \beta \cdot \Delta Y_2^W (o_{1i} \cdot \Delta Y_1^W + o_{2i} \cdot \Delta Y_2^W + \ldots\ldots + o_{di} \cdot \Delta Y_d^W) = o_{2i} \cdot \lambda_i^+ \\ \alpha\lambda \cdot o \quad + \beta \cdot \Delta Y^W (o \cdot \Delta Y^W + o \cdot \Delta Y^W + \ldots\ldots + o \cdot \Delta Y^W) = o \cdot \lambda^+ \end{cases} \qquad (1.2.12)$$

De (1.2.12), on tire:

$$\frac{o_{1i}(\lambda_1^+ - \alpha\lambda_1)}{\Delta Y_1^W} = \frac{o_{2i}(\lambda_2^+ - \alpha\lambda_2)}{\Delta Y_2^W} = \ldots\ldots = \frac{o_{di}(\lambda_i^+ - \alpha\lambda_d)}{\Delta Y_d^W} = F_i \qquad (1.2.13)$$

où $F_i = \beta(o_{1i}\cdot\Delta Y_1^W + o_{2i}\cdot\Delta Y_2^W + \ldots\ldots + o_{di}\cdot\Delta Y_d^W)$

A partir de (1.2.13), on peut écrire:

$$\lambda_i^+ = \alpha\cdot\lambda_i + \beta\cdot\Delta Y_i^W\cdot\sum_{m=1}^{d}\frac{o_{mi}}{o_{ii}}\cdot\Delta Y_m^W \qquad (1.2.14)$$

et $$\beta\cdot\left[\frac{(\Delta Y_1^W)^2}{\lambda_i^+ - \alpha\cdot\lambda_1} + \frac{(\Delta Y_2^W)^2}{\lambda_i^+ - \alpha\cdot\lambda_2} + \ldots\ldots + \frac{(\Delta Y_d^W)^2}{\lambda_i^+ - \alpha\cdot\lambda_d}\right] = 1 \qquad (1.2.15)$$

pour i=1,2,...,d

(1.2.15) conduit à la résolution d'une équation polynomiale en λ_i^+ de degré d. L'estimation des λ_i^+ se fera selon un algorithme rapide présenté dans la section suivante.

II.2 ANALYSE DE L'ALGORITHME RECURSIF RAPIDE

II.2.1 ENONCE DE L'ALGORITHME RAPIDE

Cet algorithme a été initialement proposé dans [LAKEL, 1987]. Nous en proposons, dans ce chapitre, une modification afin d'en assurer la convergence dans tous les cas et de faire apparaître toutes les solutions. Le but de l'algorithme est de trouver une résolution rapide et efficace pour l'équation (1.2.14). On peut considérer que la rotation des vecteurs propres est faible en première approximation.

soit: $o_{i\,i} \approx 1$ et $o_{j\,i} \approx 0$ pour $\forall$ i,j=1, ..., d et i≠j

Dans ces conditions, l'expression (1.2.14) fournit une première estimation pour λ_i^+: $\lambda_{i\,,\,0}^+ = \alpha \cdot \lambda_i + \beta \cdot (\Delta Y_i^W)^2$ i=1, ..., d

Ceci conduit pour λ_i^+ à la formule récursive suivante:

$$\lambda_{i\,,\,j+1}^+ = \alpha \cdot \lambda_i + \beta \cdot \Delta Y_i^W \sum_{m=1}^{d} \frac{o_{m\,i}}{o_{i\,i}} \Delta Y_m^W$$

$$= \alpha \cdot \lambda_i + \beta \cdot \Delta Y_i^W \sum_{m=1}^{d} \frac{\Delta Y_m^W}{\Delta Y_i^W} \cdot \frac{\lambda_{i\,,\,j}^+ - \alpha \cdot \lambda_i}{\lambda_{i\,,\,j}^+ - \alpha \cdot \lambda_m} \Delta Y_m^W$$

$$= \alpha \cdot \lambda_i + \beta \cdot \sum_{m=1}^{d} (\Delta Y_m^W)^2 \frac{\lambda_{i\,,\,j}^+ - \alpha \cdot \lambda_i}{\lambda_{i\,,\,j}^+ - \alpha \cdot \lambda_m} \qquad (1.2.16)$$

Cette procédure se fait récursivement jusqu'à ce que la condition ci-dessous soit satisfaite.

$\left| \lambda_{i\,,\,j+1}^+ - \lambda_{i\,,\,j}^+ \right| < \varepsilon$ où ε est la précision désirée.

II.2.2 PROPRIETES DES NOUVELLES VALEURS PROPRES

[O] est une matrice orthonormée, c'est à dire:

on a: $$\sum_{m=1}^{d} o_{mi}\, o_{mk} = \sum_{m=1}^{d} \frac{\Delta Y_m^w}{\Delta Y_i^w} \cdot \frac{\lambda_i^+ - \alpha\cdot\lambda_i}{\lambda_i^+ - \alpha\cdot\lambda_m} o_{ii} \cdot \frac{\Delta Y_m^w}{\Delta Y_k^w} \cdot \frac{\lambda_k^+ - \alpha\cdot\lambda_k}{\lambda_k^+ - \alpha\cdot\lambda_m}\, o_{kk}\,, \text{ pour } i \neq k$$

$$= 0 \qquad (1.2.17)$$

A partir de (1.2.15), on obtient:

$$\beta\cdot\left[\frac{(\Delta Y_1^w)^2}{\lambda_i^+ - \alpha\cdot\lambda_1} + \frac{(\Delta Y_2^w)^2}{\lambda_i^+ - \alpha\cdot\lambda_2} + \ldots\ldots + \frac{(\Delta Y_d^w)^2}{\lambda_i^+ - \alpha\cdot\lambda_d}\right] = 1 \qquad (1.2.18)$$

$$\beta\cdot\left[\frac{(\Delta Y_1^w)^2}{\lambda_k^+ - \alpha\cdot\lambda_1} + \frac{(\Delta Y_2^w)^2}{\lambda_k^+ - \alpha\cdot\lambda_2} + \ldots\ldots + \frac{(\Delta Y_d^w)^2}{\lambda_k^+ - \alpha\lambda_d}\right] = 1 \qquad (1.2.19)$$

En calculant (1.2.18)-(1.2.19), il vient:

$$(\lambda_i^+ - \lambda_k^+)\cdot\beta\cdot\left[\frac{(\Delta Y_1^w)^2}{(\lambda_i^+ - \alpha\cdot\lambda_1)(\lambda_k^+ - \alpha\cdot\lambda_1)} + \ldots\ldots + \frac{(\Delta Y_d^w)^2}{(\lambda_i^+ - \alpha\cdot\lambda_d)(\lambda_k^+ - \alpha\cdot\lambda_d)}\right] = 0$$

Si $\lambda_k^+ \neq \lambda_i^+$, $\sum_{m=1}^{d} \frac{(\Delta Y_m^w)^2}{(\lambda_i^+ - \alpha\cdot\lambda_m)(\lambda_k^+ - \alpha\cdot\lambda_m)} = 0$, on obtient $\sum_{m=1}^{d} o_{mi}\, o_{mk} = 0$,

et la condition (1.2.17) est satisfaite.

Si $\lambda_k^+ = \lambda_i^+$, on peut déduire de (1.2.17): $\sum_{m=1}^{d} o_{mi}\, o_{mk} \neq 0$. Dans ce cas, la condition (1.2.17) n'est pas satisfaite.

Selon l'analyse ci-dessus, la matrice [O] est orthonormée ssi λ_1^+, λ_2^+, ..., λ_d^+ sont des racines différentes du polynôme de degré d (1.2.15).

Dans la pratique, il n'existe pas de racines multiples pour l'équation (1.2.15) et il est toujours possible de trouver d racines différentes.

Dans ces conditions, il reste certains problèmes non-résolus à l'algorithme rapide:

- l'algorithme converge éventuellement vers des valeurs propres identiques (par exemple: $\lambda_i \neq \lambda_j$ mais $\lambda_i^+ = \lambda_j^+$ d'après l'algorithme).

- la convergence de l'algorithme n'est pas toujours assurée. Dans le cas non convergent, $\lambda_{i,j}^+$ boucle et il ne converge vers aucune valeur fixe.

Selon les expériences de simulation, ces deux cas apparaissent lorsque les $|\Delta Y_i^W|$ sont très grands.

II.2.3 ANALYSE DE LA CONVERGENCE DE L'ALGORITHME

La convergence de l'algorithme peut être étudiée en détail de la façon suivante:

Dans un cas particulier (d=2), la formule (1.2.16) peut se réécrire sous la forme suivante:

$$\lambda_{i,j+1}^+ = \alpha\cdot\lambda_1 + \beta\cdot(\Delta Y_1^W)^2 + \beta\cdot(\Delta Y_2^W)^2 + \alpha\cdot\beta\cdot\frac{\lambda_2-\lambda_1}{\lambda_{1,j}^+ - \alpha\cdot\lambda_2} \qquad (1.2.20)$$

En notant, pour simplifier: $x_j = \lambda_{1,j}^+$, $A_1 = \alpha\cdot\lambda_1 + \beta\cdot(\Delta Y_1^W)^2 + \beta\cdot(\Delta Y_2^W)^2$, $C_1 = \alpha\cdot\lambda_2$, $B_1 = \alpha\cdot\beta(\lambda_2-\lambda_1)$,

on obtient: $x_{j+1} = A_1 + \frac{B_1}{x_j - C_1}$ et $x_0 = \alpha\cdot\lambda_1 + \beta(\Delta Y_1^W)^2 < A_1$

Dans le plan, la solution de l'algorithme est le point d'intersection de l'hyperbole $x_{j+1} = A_1 + \frac{B_1}{x_j - C_1}$ et de la droite $x_{j+1} = x_j$. En effet, il existe 2 points d'intersection dont l'un est stable et l'autre instable. L'algorithme converge vers le point stable.

On peut analyser certains cas particuliers de la façon suivante:

1) $B_1 > 0$, $A_1 < C_1$:

$\{x_n\}$ converge vers K_1, point d'intersection de $y = x$ et $y = A_1 + \frac{B_1}{x - C_1}$.

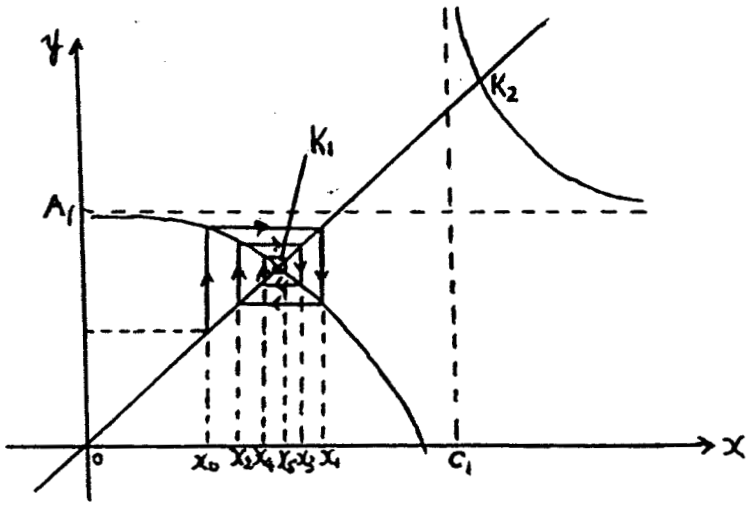


fig-1.7 déroulement l'algorithme (d=2)

[preuve]: Puisque $x_0 < A_1 < C_1$, on a: $x_1 = A_1 + \dfrac{B_1}{x_0 - C_1} < A_1$

$$\vdots$$

$$\vdots$$

$$x_n = A_1 + \frac{B_1}{x_{n-1} - C_1} < A_1$$

Si $x_{n-1} > x_n$, on obtient: $x_n - C_1 < x_{n-1} - C_1 < 0$

Donc, $A_1 + \dfrac{B_1}{x_n - C_1} > \dfrac{B_1}{x_{n-1} - C_1} + A_1$, soit: $x_{n+1} > x_n$

A partir de $x_n < x_{n+1}$, on obtient $x_{n+2} < x_{n+1}$

On conclut que $A_1 + \dfrac{B_1}{x_{n+1} - C_1} < x_{n+1}$ si $A_1 + \dfrac{B_1}{x_{n-1} - C_1} < x_{n-1}$

De la même façon, on obtient:

$$A_1 + \frac{B_1}{x_{n+1} - C_1} > x_{n+1} \quad \text{si } A_1 + \frac{B_1}{x_{n-1} - C_1} > x_{n-1}$$

$$\text{si } A_1 + \frac{B_1}{x_{n-1} - C_1} > x_{n-1} \text{ , on a: } A_1 + \frac{B_1}{x_n - C_1} < x_n$$

$$\text{si } A_1 + \frac{B_1}{x_{n-1} - C_1} < x_{n-1} \text{ , on a: } A_1 + \frac{B_1}{x_n - C_1} > x_n$$

Donc, x_0 , x_2 , x_4 , ..., x_{2n} se trouvent d'un coté de K_1 ,

et x_1, x_3, ..., x_{2n+1} se trouvent de l'autre coté de K_1.

Ensuite, on essaie de prouver que $\{x_{2n}\}$ et $\{x_{2n-1}\}$ sont monotones respectivement.

De toute façon, on a: $K_1 < C_1 < K_2$.

On considère la séquence $\{x_{n+1}, x_{n-1}, ...\}$ à gauche de K_1.

A partir de $x_{n+1} > x_{n-1}$, on obtient ci-dessous les conditions necessaires et suffisantes:

$$A_1 + \frac{B_1}{x_n - C_1} > x_{n-1}$$

$$A_1 (x_n - C_1) + B_1 < x_{n-1} (x_n - C_1)$$

$$A_1 \left(\frac{B_1}{x_{n-1} - C_1} + A_1 - C_1\right) + B_1 < x_{n-1} \left(A_1 + \frac{B_1}{x - C_1} - C_1\right)$$

$$A_1^2 (x_{n-1} - C_1) + A_1 B_1 - A_1 C_1 (x_{n-1} - C_1) + B_1 (x_{n-1} - C_1)$$
$$> x_{n-1} (A_1 - C_1)(x_{n-1} - C_1) + B_1 x_{n-1}$$

$$f(x_{n-1}) = (C_1 - A_1) x_{n-1}^2 + A_1^2 x_{n-1} - A_1^2 C_1 - B_1 (C_1 - A_1) > 0 \qquad (1.2.21)$$

où $C_1 - A_1 > 0$, $A_1^2 > 0$

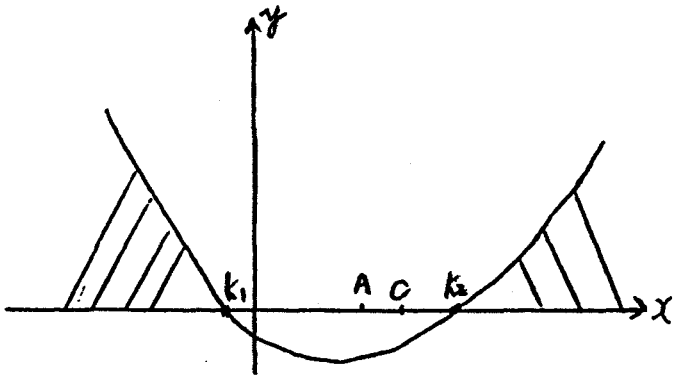


fig-1.8 parabole relative à l'équation (1.2.21)

Selon la fig-1.8, on a: $x_{n+1} > x_{n-1}$ ssi $x_{n-1} < K_1$ ou $x_{n-1} > K_2$. Lorsque cette séquence se situe à gauche de K_1, on a $x_{n-1} < K_1$. Evidemment, $x_{n+1} > x_{n-1}$ et la séquence $\{x_{n+1}, x_{n-1}, ...\}$ est croissante.

De la même façon, on peut conclure que la séquence à droite de K_1: $\{x_{n+1}, x_{n-1}, ...\}$ est décroissante. Donc, ces deux séquences sont toutes deux convergentes.

De plus, possédant la même forme, elles convergent vers la même valeur K_1.

Conclusion: si $B_1 > 0$, $A_1 < C_1$, la séquence $\{x_n\}$ converge vers K_1.

2) De la même manière, on peut prouver la convergence de $\{x_n\}$ vers K_1 ou K_2 dans les autres cas.

Pour l'autre valeur propre λ_2^+, la formule (1.2.20) se réécrit comme suit:

$$\lambda^+_{2,j+1} = A_2 + \frac{B_2}{\lambda^+_{2,j} - C_2} \quad \text{où} \quad A_2 = \alpha\cdot\lambda_2 + \beta(\Delta X_1^w)^2 + \beta(\Delta X_2^w)^2$$

$$B_2 = \alpha\cdot\beta(\Delta X_1^w)^2(\lambda_1 - \lambda_2)$$
$$C_2 = \alpha\cdot\lambda_1$$

Nous pouvons déduire les conditions suivantes:

$$A_1 > C_2, \quad A_2 > C_1 \quad \text{et} \quad B_1 B_2 < 0$$

En résumé, on définit la convergence pour tous les cas:

1) $\begin{cases} B_1 > 0 \\ A_1 < C_1 \end{cases}$, on obtient $\begin{cases} B_2 < 0 \\ A_2 > C_2 \end{cases}$ et $\begin{cases} \lambda_1^+ \to K_1 \\ \lambda_2^+ \to K_2 \end{cases}$ (1.2.22a)

2) $\begin{cases} B_1 > 0 \\ A_1 > C_1 \end{cases}$, on obtient $\begin{cases} B_2 < 0 \\ A_2 > C_2 \end{cases}$ et $\begin{cases} \lambda_1^+ \to K_2 \\ \lambda_2^+ \to K_2 \end{cases}$ (1.2.22b)

ou $\begin{cases} B_2 < 0 \\ C_2 > A_2 \end{cases}$ et $\begin{cases} \lambda_1^+ \to K_2 \\ \lambda_2^+ \to K_1 \end{cases}$ (1.2.22c)

3) $\begin{cases} B_1 < 0 \\ A_1 > C_1 \end{cases}$, on obtient $\begin{cases} B_2 > 0 \\ A_2 > C_2 \end{cases}$ et $\begin{cases} \lambda_1^+ \to K_2 \\ \lambda_2^+ \to K_2 \end{cases}$ (1.2.22d)

ou $\begin{cases} B_2 > 0 \\ A_2 < C_2 \end{cases}$ et $\begin{cases} \lambda_1^+ \to K_1 \\ \lambda_2^+ \to K_2 \end{cases}$ (1.2.22e)

$$4)\ \begin{cases} B_1 > 0 \\ A_1 < C_1 \end{cases},\ \text{on obtient}\ \begin{cases} B_2 > 0 \\ C_2 < A_2 \end{cases}\ \text{et}\ \begin{cases} \lambda_1^+ \to K_1 \\ \lambda_2^+ \to K_2 \end{cases} \qquad (1.2.22f)$$

Donc, (1.2.22b) et (1.2.22d) sont les deux cas où l'algorithme conduit λ_i^+ à converger vers deux valeurs identiques et on ne peut pas obtenir d racines différentes d'après cet algorithme.

Si d>2, l'étude théorique de la convergence est trop complexe. Mais les résultats sont similaires. L'algorithme ne peut donc pas assurer la convergence des valeurs propres vers d racines différentes de l'équation (1.2.15).

II.3 ALGORITHME RECURSIF RAPIDE MODIFIE

II.3.1 ENONCE DE L'ALGORITHME RAPIDE MODIFIE

Pour calculer les λ_i^+ différentes, on modifie l'algorithme de la façon suivante:

A l'initialisation, on note $\Delta Z_m = (\Delta Y_m^w)^2$ pour m=1, 2, ..., d.

Pour $\forall$ i=1, ..., d, on exécute la procédure récursive en parallèle:

$$\lambda_{i,j+1}^+ = \alpha\cdot\lambda_i + \beta\cdot\sum_{m=1}^{d} \frac{(\Delta Z_m)(\lambda_{i,j}^+ - \alpha\cdot\lambda_i)}{\lambda_{i,j}^+ - \alpha\cdot\lambda_m} \qquad (1.2.23)$$

Supposons qu'on obtienne k (k<d) valeurs différentes: λ_1^+, λ_2^+, ..., λ_k^+ à partir de $\{\lambda_i\}$ (i=1,2,...,d).

Pour une racine quelconque non-résolue λ_i^+ (i>k), l'équation de degré d (1.2.15) est vérifiée. C'est à dire:

$$\beta\cdot\left[\frac{(\Delta Z_1)}{\lambda_i^+ - \alpha\cdot\lambda_1} + \frac{(\Delta Z_2)}{\lambda_i^+ - \alpha\cdot\lambda_2} + \cdots\cdots + \frac{(\Delta Z_d)}{\lambda_i^+ - \alpha\cdot\lambda_d}\right] = 1 \qquad (1.2.24)$$

A partir de (1.2.24), on a:

$$\lambda_i^+ = \alpha\cdot\lambda_i + \beta\cdot\sum_{m=1}^{d} \frac{(\Delta Z_m)(\lambda_i^+ - \alpha\cdot\lambda_i)}{\lambda_i^+ - \alpha\cdot\lambda_m} \qquad (1.2.25)$$

En même temps, λ_1^+ satisfait aussi à l'équation suivante:

$$\lambda_1^+ = \alpha\cdot\lambda_i + \beta\cdot\sum_{m=1}^{d} \frac{(\Delta Z_m)(\lambda_1^+ - \alpha\cdot\lambda_i)}{\lambda_1^+ - \alpha\cdot\lambda_m} \qquad (1.2.26)$$

En effectuant (1.2.25) - (1.2.26), il vient:

$$\lambda_i^+ - \lambda_1^+ = \beta\cdot\sum_{m=1}^{d} \frac{(\Delta Z_m)(\lambda_i^+ - \lambda_1^+)(\lambda_1 - \lambda_m)}{(\lambda_i^+ - \alpha\cdot\lambda_m)(\lambda_1^+ - \alpha\cdot\lambda_m)}$$

On obtient une nouvelle équation de degré (d-1):

$$\beta \cdot \left[\frac{(\Delta Z_2)}{\lambda_i^+ - \alpha \cdot \lambda_2} + \frac{(\Delta Z_3)}{\lambda_i^+ - \alpha \cdot \lambda_3} + \ldots\ldots + \frac{(\Delta Z_d)}{\lambda_i^+ - \alpha \cdot \lambda_d} \right] = 1 \qquad (1.2.27)$$

$$\text{où } (\Delta Z_m) := \frac{\lambda_1 - \lambda_m}{\lambda_1^+ - \alpha \cdot \lambda_m} \cdot (\Delta Z_m), \quad m=2, 3, \ldots, d$$

A partir de cette équation, on répète la procédure ci-dessus pour λ_2, λ_3, ..., λ_k. Finalement, on obtient l'équation de degré (d-k) suivante:

$$\beta \cdot \left[\frac{(\Delta Z_{k+1})}{\lambda_i^+ - \alpha \cdot \lambda_{k+1}} + \frac{(\Delta Z_{k+2})}{\lambda_i^+ - \alpha \cdot \lambda_{k+2}} + \ldots\ldots + \frac{(\Delta Z_d)}{\lambda_i^+ - \alpha \cdot \lambda_d} \right] = 1 \qquad (1.2.28)$$

$$\text{où } (\Delta Z_m) := \frac{\prod_{j=1}^{k} (\lambda_j - \lambda_m)}{\prod_{j=1}^{k} (\lambda_j^+ - \alpha \cdot \lambda_m)} \cdot (\Delta Z_m), \quad m=k+1, \ldots\ldots, d$$

A partir de l'équation (1.2.28), on exécute l'algorithme rapide une 2ème fois. Cette procédure se termine jusqu'à l'obtention des d racines différentes de (1.2.15).

II.3.2 LIMITES DE L'ALGORITHME RAPIDE

Lorsque d>2, 2 inconvénients supplémentaires subsistent dans l'algorithme rapide:

1) La valeur initiale ne conduit pas forcément l'algorithme vers un point stable.

Généralement, parmi les points d'intersection de la droite $\lambda_{i,j+1}^+ = \lambda_{i,j}^+$ et de la courbe $\lambda_{i,j+1}^+ = \alpha \cdot \lambda_i + \sum_{m=1}^{d} \frac{(\Delta Z_m)}{\lambda_{i,j}^+ - \alpha \cdot \lambda_m}$, il existe un seul point stable et les autres sont instables. Si la valeur initiale se trouve dans les zones de points instables et s'éloigne du point stable, il est possible que l'algorithme ne converge pas vers le point stable. La valeur de $\lambda_{i,j}^+$ boucle entre les points instables.

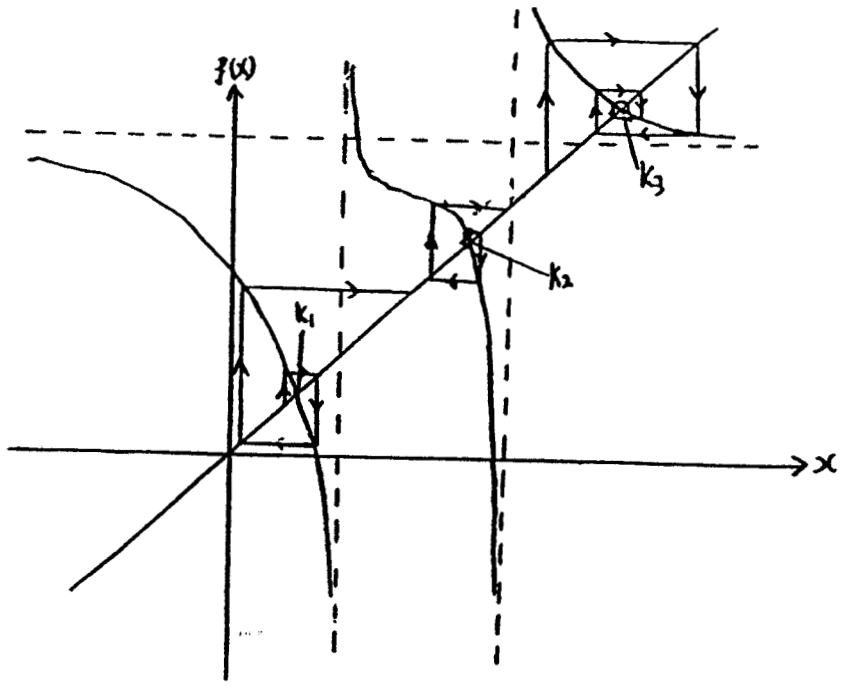


fig-1.9 déroulement de l'algorithme (d=3)

Dans la fig-1.9, K_3 est un point stable et K_1, K_2 sont instables. Si on choisit la valeur initiale au voisinage de K_1 ou K_2, la convergence n'est pas assurée. Mais si la valeur initiale se trouve au voisinage de K_3, l'algorithme converge vers K_3.

Pour assurer la convergence de cet algorithme, on propose une méthode de sélection de la valeur initiale:

Nous avons d valeurs initiales à définir:

$$d\lambda^+_{1,0} = \alpha\cdot\lambda_1 + \beta\cdot(\Delta Y^w_1)^2,\ d\lambda^+_{2,0} = \alpha\cdot\lambda_2 + \beta\cdot(\Delta Y^w_2)^2,\ \ldots\ldots,$$
$$d\lambda^+_{d,0} = \alpha\cdot\lambda_d + \beta\cdot(\Delta Y^w_d)^2$$

Chaque $d\lambda^+_{j,0}$ (j=1, ..., d) est reporté dans la formule (1.2.23) en assignant $\lambda^+_{i,0} = d\lambda^+_{j,0}$.

Donc, il vient:

$$\begin{cases} \lambda^+_{i,1} = \alpha\cdot\lambda_i + \beta\cdot\sum_{m=1}^{d} \dfrac{(\Delta Z_m)(\lambda^+_{i,0} - \alpha\cdot\lambda_i)}{\lambda^+_{i,0} - \alpha\cdot\lambda_m} \\ \lambda^+_{i,2} = \alpha\cdot\lambda_i + \beta\cdot\sum_{m=1}^{d} \dfrac{(\Delta Z_m)(\lambda^+_{i,1} - \alpha\cdot\lambda_i)}{\lambda^+_{i,1} - \alpha\cdot\lambda_m} \end{cases} \qquad (1.2.29)$$

On obtient:

$$\lambda^+_{i,2} - \lambda^+_{i,1} = \beta\cdot\sum_{m=1}^{d} \frac{-\Delta Z_m\cdot\alpha\cdot(\lambda_m - \lambda_i)(\lambda^+_{i,1} - \lambda^+_{i,0})}{(\lambda^+_{i,0} - \alpha\cdot\lambda_m)(\lambda^+_{i,1} - \alpha\cdot\lambda_m)}$$

$$\text{et } \delta = \left| \frac{\lambda^+_{i,2} - \lambda^+_{i,1}}{\lambda^+_{i,1} - \lambda^+_{i,0}} \right|$$

$$= \alpha \cdot \beta \cdot \left| \sum_{m=1}^{d} \frac{\Delta Z_m (\lambda_m - \lambda_i)}{(\lambda^+_{i,0} - \alpha \cdot \lambda_m)(\lambda^+_{i,1} - \alpha \cdot \lambda_m)} \right| \qquad (1.2.30)$$

Supposons que $\lambda^+_{i,0}$ se trouve entre deux valeurs proches $\alpha \cdot \lambda_k$, $\alpha \cdot \lambda_{k+1}$ (c'est à dire: $\lambda^+_{i,0} \in]\alpha \cdot \lambda_k, \alpha \cdot \lambda_{k+1}[$).

Pour chaque valeur de $d\lambda^+_{j,0}$ (j = 1, ..., d), on teste si $\delta < 1$ et $\alpha \cdot \lambda_k < \lambda^+_{i,1} < \alpha \cdot \lambda_{k+1}$. On sélectionne un $d\lambda^+_{j,0}$ satisfaisant à cette condition comme la valeur initiale de λ^+_i. Normalement, cette valeur initiale est assez proche du point stable et conduit l'algorithme vers ce point.

2) tous les points d'intersections sont instables:

Dans ce cas-là, la formule ne converge vers aucune valeur pour i=1, 2, ..., d. L'équation originale ne peut pas être résolue par la méthode de l'algorithme rapide.

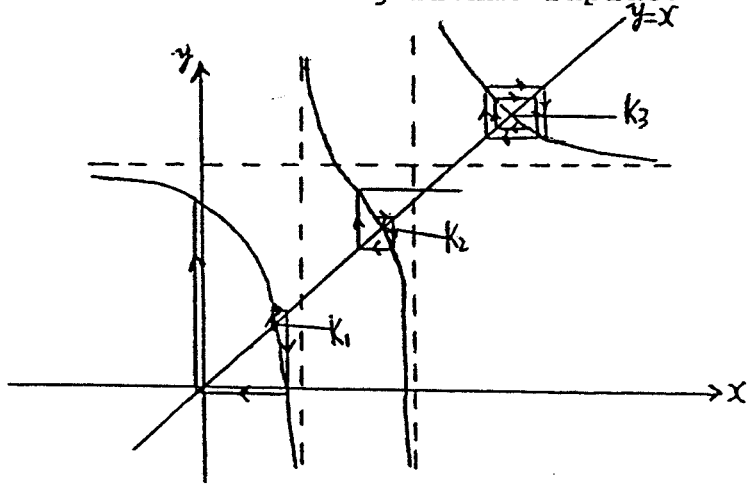


fig-1.10 déroulement de l'algorithme
(K_1, K_2, K_3 sont tous instables)

Dans cette situation, nous sommes obligés d'utiliser les algorithmes A_1 ou A_2 définis ci-dessous. Après la résolution de A_1 ou A_2, on retourne à l'algorithme rapide pour calculer les valeurs propres suivantes et ainsi de suite.

II.3.3 ENONCE DES ALGORITHMES A1 ET A2

A partir de l'équation originale (1.2.15), on obtient:

$$\lambda_i^+ = \alpha\cdot\lambda_1 + \sum_{m=1}^{d} \beta\cdot(\Delta Y_m^u)^2 + \alpha\cdot\beta\cdot\sum_{m=1}^{d} \frac{(\lambda_m - \lambda_1)(\Delta Y_m^u)^2}{\lambda_i^+ - \alpha\cdot\lambda_m} \qquad (1.2.31)$$

On ordonne toutes les valeurs propres λ_i afin que $0 < \lambda_1 < \lambda_2 < \ldots\ldots < \lambda_d$.

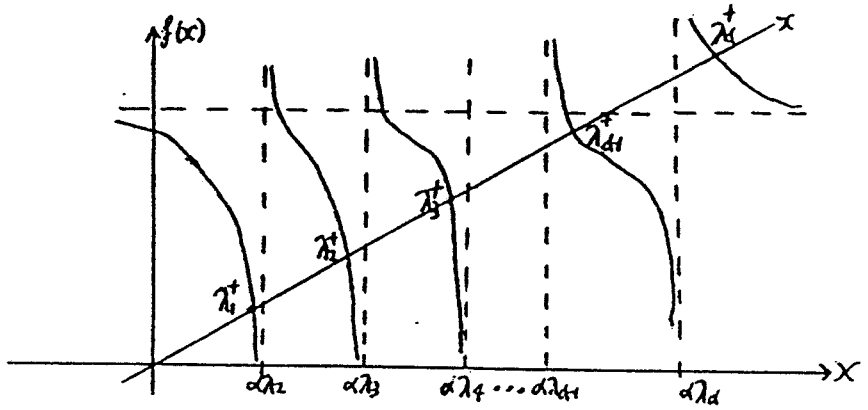


fig-1.11 solutions de l'équation (1.2.31)

Evidemment, il existe d racines différentes dans cette équation (1.2.31) et chaque racine se trouve dans l'intervale: $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_{k+1}[$. Par conséquent, $]-\infty, \alpha\cdot\lambda_2[$, $]\alpha\cdot\lambda_2, \alpha\cdot\lambda_3[$, ..., $]\alpha\cdot\lambda_{d-1}, \alpha\cdot\lambda_d[$, $]\alpha\cdot\lambda_d, +\infty[$ comportent toutes les racines λ_i^+ (i=1, ..., d). On peut les résoudre en parallèle.

Pour chaque intervalle $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_{k+1}[$, on propose la valeur initiale $\lambda_{k,0}^+ = (\alpha\cdot\lambda_k + \alpha\cdot\lambda_{k+1})/2$.

$$\text{Soit } f(x) = \alpha\cdot\lambda_1 + \sum_{m=1}^{d} \beta\cdot(\Delta Y_m^u)^2 + \alpha\cdot\beta\cdot\sum_{m=1}^{d} \frac{(\lambda_m - \lambda_1)(\Delta Y_m^u)^2}{x - \alpha\cdot\lambda_m}$$

$$= A + \sum_{m=2}^{d} \frac{B_m}{x - \alpha\cdot\lambda_m} \quad \text{où } A>0,\ B_m>0 \text{ pour } \forall\ m=2, \ldots, d$$

$\begin{cases} x = f(x) \\ \alpha\cdot\lambda_k < x < \alpha\cdot\lambda_{k+1} \end{cases}$ est la solution souhaitée, elle pourra se calculer à partir de l'algoritme A_1 ou A_2.

Algorithme A1:

On prend le milieu de l'intervalle $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_{k+1}[$,

$$x_0 = \frac{1}{2}(\alpha\cdot\lambda_{k+1} + \alpha\cdot\lambda_k) = \frac{1}{2}\cdot 1 + \alpha\cdot\lambda_k \qquad \text{où} \quad 1 = \alpha\cdot\lambda_{k+1} - \alpha\cdot\lambda_k$$

si $x_0 > f(x_0)$, alors $x_1 = \alpha\cdot\lambda_k + \frac{1}{4}\cdot 1 \in]\alpha\cdot\lambda_k, x_0[$

si $x_0 < f(x_0)$, alors $x_1 = \alpha\cdot\lambda_{k+1} - \frac{1}{4}\cdot 1 \in]x_0, \alpha\cdot\lambda_{k+1}[$

Cette procédure se répète jusqu'à $|x_n - f(x_n)| < \delta$, (δ est la précision désirée).

La longueur de l'intervale $l_n = |x_n - x_{n-1}| < \frac{1}{2^{n+1}} = \epsilon$

Le nombre d'itérations est $k = n+1 = \log_2 \frac{1}{\epsilon}$.

Algorithme A2:(fig-1.12)

Dans la plupart des cas, au voisinage du point d'intersection K, la courbe y=f(x) et la droite y=x sont similaires aux deux triangles ΔABK et ΔCDK. Si nous construisons deux triangles en utilisant respectivement deux points sur la droite et deux points sur la courbe, le point d'intersection de ces 2 triangles

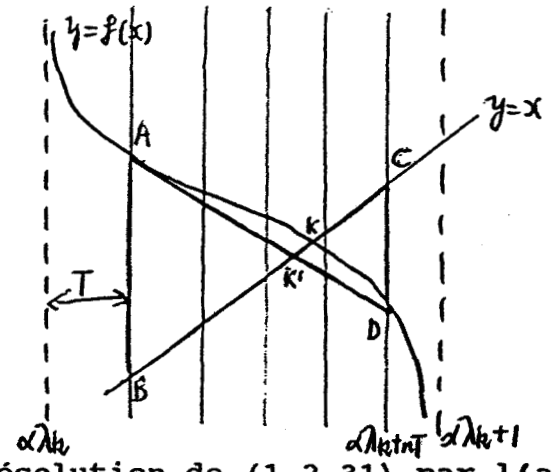


fig-1.12 résolution de (1.2.31) par l'algorithme A_2

est assez proche de K. On répète cette procédure avec ce point et on obtient une séquence de points d'intersection qui tend vers K. Basé sur la propriété ci-dessus, A_2 s'écrit de la façon suivante:

étape 1: l'intervalle $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_{k+1}[$ est divisé en (n+1) sous intervalles (fig-1.12).

étape 2: si $f(\alpha\cdot\lambda_k + T) < \alpha\cdot\lambda_k + T$, alors on considère $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_k + T[$ comme intervalle à l'itération suivante.

si $f(\alpha\cdot\lambda_k + nT) > \alpha\cdot\lambda_k + nT$, $]\alpha\cdot\lambda_k + nT, \alpha\cdot\lambda_{k+1}[$ est considéré comme intervalle à l'itération suivante.

si $f(\alpha\cdot\lambda_k + T) > \alpha\cdot\lambda_k + T$ et $f(\alpha\cdot\lambda_k + nT) < \alpha\cdot\lambda_k + nT$, car $f(x)$ est quasi linéaire dans $]z_1, z_2[$ $(z_1 = \alpha\cdot\lambda_k + T,\ z_2 = \alpha\cdot\lambda_k + nT)$,

on a: $\dfrac{K' - z_1}{-K' + z_2} = \dfrac{y_1}{y_2}$ où $\begin{cases} y_1 = f(\alpha\cdot\lambda_k + T) - (\alpha\cdot\lambda_k + T) \\ y_2 = \alpha\cdot\lambda_k + nT - f(\alpha\cdot\lambda_k + nT) \end{cases}$

il vient: $K' = \dfrac{z_2 y_1 + z_1 y_2}{y_1 + y_2}$

K' est le premier point d'intersection des triangles

Si $|K' - f(K')| < \delta$, l'algorithme s'arrête et $K := K'$; sinon, on envisage la relation entre K' et $f(K')$.

Dans beaucoup de cas, K' est assez proche de K et K se trouve souvent dans l'intervalle $]K' - T, K' + T[$.

Si $K' < f(K')$, alors on prend $]K', K' + T[$ comme intervalle suivant lorsque $K' + T > f(K' + T)$ et on prend $]K', z_2[$ comme intervalle suivant lorsque $K' - T < f(K' - T)$.

Si $K' > f(K')$, alors on considère $]K' - T, K'[$ comme intervalle suivant lorsque $K' - T < f(K' - T)$ et on considère $]z_1, K'[$ comme intervalle suivant lorsque $K' - T > f(K' - T)$.

Ensuite, on retourne à l'étape 1 et on répète cette procédure jusqu'à l'obtention de K.

Evidemment, A_1 et A_2 sont parallélisables. Chaque processeur peut calculer la nouvelle valeur propre dans un intervalle $]\alpha\cdot\lambda_k, \alpha\cdot\lambda_{k+1}[$. Pourtant, la vitesse de convergence de A_2 est plus rapide que celle de A_1.

En effet, supposons $l = \alpha\cdot\lambda_{k+1} - \alpha\cdot\lambda_k$, dans le cas optimal de A_2, on a: $\dfrac{l}{n^{k'}} < \epsilon$ et le nombre d'itérations est $k' = \log_n \dfrac{l}{\epsilon}$. Dans A_1, le nombre d'itérations $k = \log_2 \dfrac{l}{\epsilon}$.

il vient: $\dfrac{k}{k'} = \log_2 n$

Par conséquent, l'algorithme A_2 est plus rapide et plus efficace que A_1. Le choix de n est très important dans A_2. Si n est trop grand, $k \notin]K'-T, K'+T[$. Si n est trop petit, la vitesse de convergence est beaucoup moins rapide. Dans la simulation de A_2, on choisit n=10.

II.3.4 MISE EN PARALLELE DES ALGORITHMES

Les algorithmes présentés ci-dessus possèdent l'avantage d'être facilement parallélisables.

En effet, selon l'algorithme rapide, le calcul de $\{\lambda_i^+\}$ (i=1,...,d) est basé sur la formule récursive suivante:

$$\lambda_{i,j+1}^+ = \alpha \cdot \lambda_{i,j}^+ + \beta \cdot \sum_{m=q}^{d} \frac{(\Delta Z_m)(\lambda_{i,j}^+ - \alpha\lambda_i)}{\lambda_{i,j}^+ - \alpha\lambda_m} \qquad (1.2.32)$$

où q est un entier positif et $1 \leqslant q \leqslant d$

On donne dans la fig-1.13 le schéma du déroulement lorsqu'on intégre un nouvel échantillon Y_{n+1}:

Il apparaît que le calculs des éléments propres relatifs à un i donné peuvent être effectués de manière indépendante. Par conséquent, il est possible d'effectuer les calculs avec d processeurs. Ainsi, le temps d'intégration d'un nouvel échantillon est divisé par d.

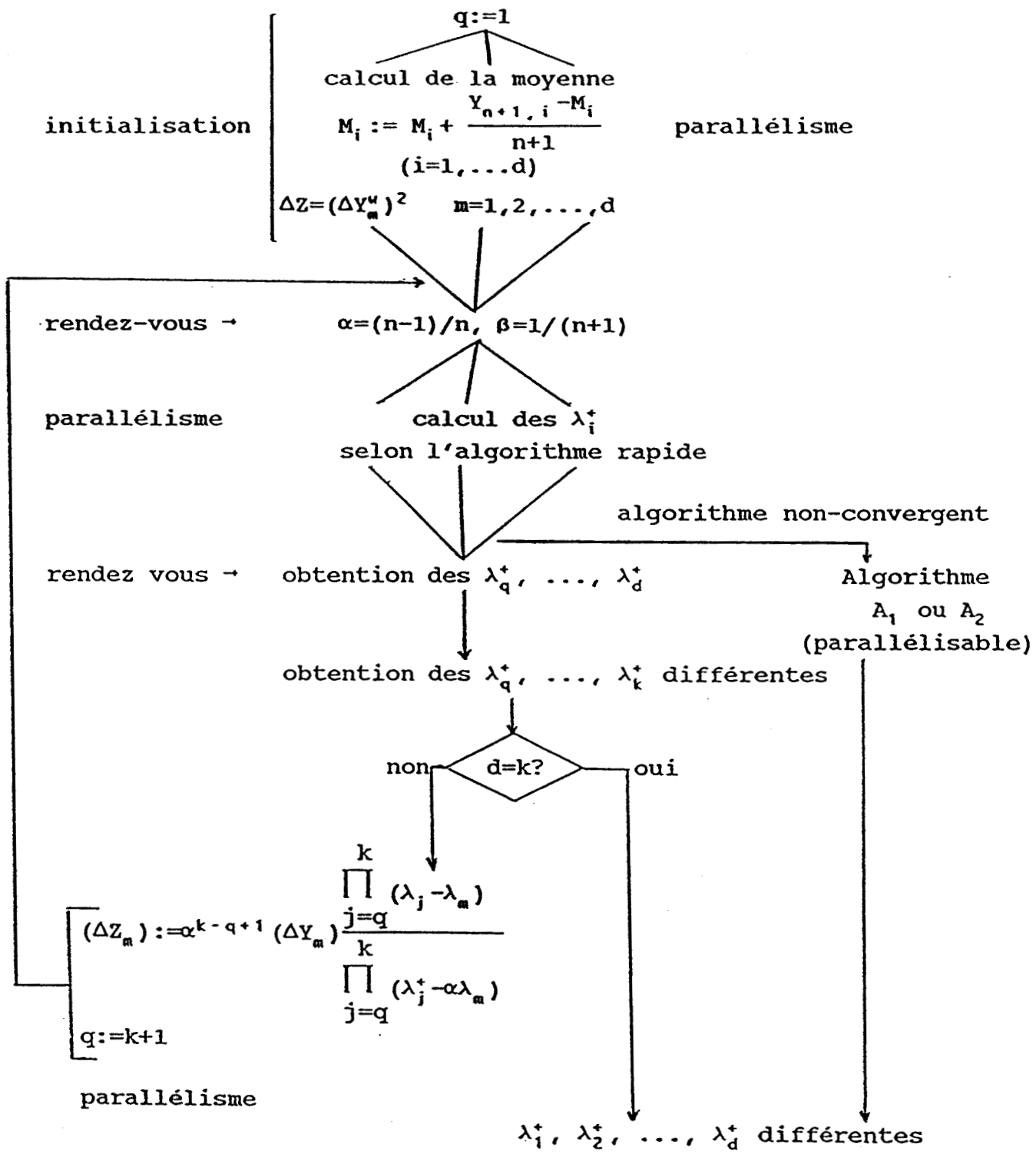


fig-1.13 étude du parallélisme

II.4 GLISSEMENT D'UNE FENETRE TEMPORELLE PARMI UN MELANGE DE CLASSES

II.4.1 PRINCIPE DU GLISSEMENT DE FENETRE

Au cours de l'évolution temporelle des échantillons, on élabore une fenêtre d'observation pour caractériser les paramètres statistiques de ces échantillons et rendre compte du changement des valeurs caractéristiques parmi plusieurs classes.

Le glissement de fenêtre se déroule de la manière suivante:

1) On prend n échantillons comme base d'apprentissage, ce qui constitue la fenêtre initiale.

2) Un nouvel échantillon est intégré à la fenêtre actuelle. Cette étape, appelée récursivité positive et développée dans II.3, consiste à estimer les paramètres statistiques de la classe après ajout d'un échantillon.

3) Le plus ancien échantillon de la fenêtre est retiré. Cette étape, appelée récursivité négative, est développé dans les paragraphes suivants. Elle consiste à estimer les paramètres statistiques après retrait de cet échantillon.

4) On répète 2) et 3), ce qui réalise le glissement de fenêtre.

II.4.2 RECURSIVITE NEGATIVE

Par la même méthode que la récursivité positive, on obtient:

$$S'=\alpha'\cdot S + \beta'\cdot R \quad \text{avec } \alpha'= \frac{n-1}{n-2},\ \beta'= \frac{-n}{(n-1)(n-2)}.$$

$R = (Y_n - M)(Y_n - M)^T$ possédant la même forme que dans (1.2.9).

S' est l'estimation de la matrice de covariance après retrait d'un échantillon.

Dans cette étape, le calcul des nouvelles valeurs propres et vecteurs propres est identique à celui de la récursivité positive présenté ci-dessus. Il suffit, pour les obtenir, de

remplacer α par α' et β par β' dans les formules correspondantes.

Ensuite, on analyse la variation des paramètres caractéristiques au cours de l'évolution des échantillons du mélange de classes.

II.4.3 EVOLUTION TEMPORELLE DES PARAMETRES

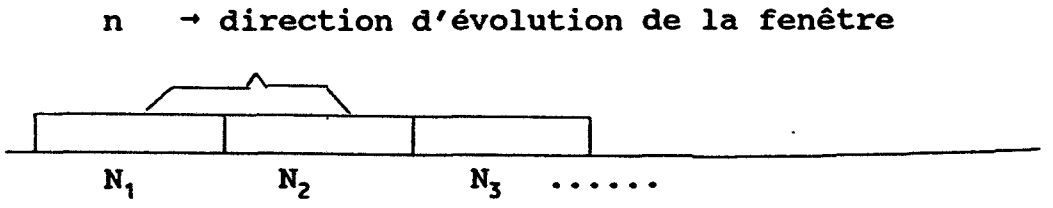


fig-1.14 glissement de fenêtre parmi plusieurs classes

Supposons la taille de fenêtre $n<N_i$ (N_i est le nombre d'échantillons dans la classe C_i). On envisage un cas où la fenêtre pénètre dans une nouvelle classe.

Nous pouvons faire les remarques suivantes:

Le nombre d'échantillons de la nouvelle classe appartenant à cette fenêtre est $i \leqslant n$.

Le nombre d'échantillons de l'ancienne classe appartenant à cette fenêtre est n-i.

Selon la formule (1.2.3), l'estimation du vecteur moyenne est:

$$M(i) = \frac{1}{n} \sum_{j=1}^{n} Y_j$$

L'espérance mathématique de cette estimation s'écrit de la façon suivante:

$$E(M(i)) = \frac{1}{n} \sum_{j=1}^{n} E(Y_j) = \frac{1}{n}((n-i)\cdot M_1 + i\cdot M_2) = M_1 + \frac{1}{n}(M_2 - M_1)i \qquad (1.2.33)$$

où M_1, M_2 sont respectivement les vecteurs moyennes des deux classes.

De la même manière, on peut déduire l'estimation de la

matrice de covariance de la fenêtre S(i):

$$S(i) = \frac{1}{n-1}\sum_{j=1}^{n}(Y_j - M(i))(Y_j - M(i))^T$$

Son espérance mathématique est:

$$E(S(i)) = \frac{1}{n-1}\sum_{i=1}^{n} E((Y_i - M(i))(Y_i - M(i))^T) \qquad (1.2.34)$$

Soient S_1, S_2 respectivement les matrices de covariance de ces deux classes, on obtient les résultats suivants:

Lorsque Y_j appartient à l'ancienne classe,

$$A(i) = E((Y_j - M(i))(Y_j - M(i))^T)$$
$$= \frac{1}{n^2}[-i^2 M_2 M_1^T - i^2 M_1 M_2^T + i^2 M_2 M_2^T + i^2 M_1 M_1^T + (n^2 - n - 1)S_1 + i \cdot S_2]$$

Lorsque Y_j appartient à la nouvelle classe,

$$B(i) = E((Y_j - M(i))(Y_j - M(i))^T)$$
$$= \frac{1}{n^2}[(n-i)S_1 + (n^2 - 2n + i)S_2 + (n-i)^2 M_1 M_1^T + (n^2 + i^2 - 2n \cdot i)M_2 M_2^T$$
$$-(n-i)^2 M_1 M_2^T - (n-i)^2 M_2 M_1^T]$$

On en déduit:

$$E(S(i)) = \frac{1}{n-1}(i \cdot B(i) + (n-i) \cdot A(i))$$
$$= \frac{n-i}{n}S_1 + \frac{i}{n}S_2 + \frac{1}{n(n-1)}[(n-i)i \cdot M_1 M_1^T + (n-i)i \cdot M_2 M_2^T$$
$$-(n-i)i \cdot M_1 M_2^T - (n-i)i \cdot M_2 M_1^T]$$

On obtient donc les conclusions suivantes:

$$E(M(i)) = M_1 + \frac{1}{n}(M_2 - M_1)i \qquad (1.2.35)$$

$$E(S(i)) = S_1 + \frac{S_2 - S_1}{n}i + \frac{i(n-i)}{n(n-1)}(M_1 M_1^T + M_2 M_2^T - M_1 M_2^T - M_2 M_1^T) \qquad (1.2.36)$$

On peut discuter certains cas particuliers:

1) i=0: $E(M(i))=M_1$ et $E(S(i))=S_1$,
la fenêtre est immergée dans l'ancienne classe

2) i=n: $E(M(i))=M_2$ et $E(S(i))=S_2$,
la fenêtre est immergée dans la nouvelle classe

3) $i=\frac{n}{2}$ (si n est pair): $E(M(i)) = \frac{M_1+M_2}{2}$

$$E(S(i)) = \frac{S_1+S_2}{2} + \frac{n}{4(n-1)}(M_1 M_1^I + M_2 M_2^I - M_1 M_2^I - M_2 M_1^I)$$

la fenêtre se trouve au milieu des 2 classes.

II.4.4 CAS PARTICULIER D'UN MELANGE DE CLASSES ELOIGNEES

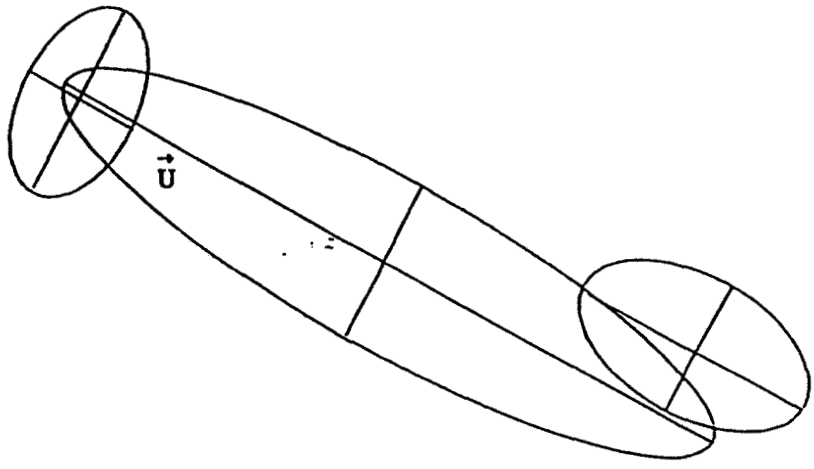


fig-1.15 nuage des échantillons entre 2 classes

Lorsque les deux classes sont assez éloignées, nous pouvons énoncer la propriété suivante:

Au cours de l'évolution de la fenêtre entre 2 classes, il existe une forte direction $\vec{U}$. La valeur propre de S(i), qui correspond à cette direction, est beaucoup plus grande que les autres. $\vec{U}$ est quasiment constant pendant l'évolution de la fenêtre. Cette propriété est vérifiée par les expériences de simulation.

II.5 RESULTATS DE SIMULATION

II.5.1 PRINCIPE DE LA SIMULATION

Le but de la simulation est de tester la rapidité, l'efficacité et les performances des algorithmes proposés lorsqu'on intégre un à un des échantillons qui viennent de classes différentes.

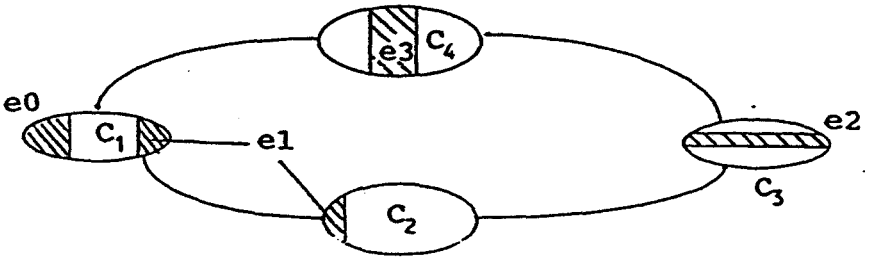
Le schéma de travail est le suivant:

ETAPE 1: Génération artificielle des différentes classes C_i contenant respectivement N_i échantillons (i=1,...,m) pour chaque classe. Ces classes sont caractérisées par les paramètres statistiques: vecteur moyenne, valeurs et vecteurs propres de la matrice de covariance.

ETAPE 2: Extraction à partir de la classe C_1 d'un sous ensemble e0 de n échantillons par tirage aléatoire ($n \leq N_i$ pour i=1,...m). e0 constitue la fenêtre initiale.

ETAPE 3: Calcul des paramètres statistiques de e0 par diagonalisation de la matrice de covariance.

ETAPE 4: Intégrations successives dans la fenêtre des échantillons restant dans C_1 puis des échantillons dans C_i (i=2,...,m). La fenêtre traverse successivement les classes C_1, C_2, ..., C_m, C_1 et on s'arrête au point de départ dans C_1. L'estimation instantanée des nouveaux paramètres statistiques au cours de l'évolution de la fenêtre de valeur initiale e0 est effectuée par l'algorithme rapide, sa modification et l'algorithme A_2.



où e0, e1, e2, e3 sont 4 images instantanées de la fenêtre

fig-1.16 trajectoire d'une fenêtre à travers plusieurs classes

II.5.2 RESULTATS OBTENUS

Dans la simulation, on choisit une taille de fenêtre n=30. Le tableau-1.1 montre les résultats de l'évolution des paramètres (vecteur moyenne et matrice de covariance), dans le cas de deux classes pour d=2.

i	$M_1(i)$	$M_2(i)$	$S_{11}(i)$	$S_{12}(i)$	$S_{21}(i)$	$S_{22}(i)$
0	2.380	3.972	0.24	0.05	0.05	0.16
1	2.724	4.418	3.76	4.61	4.61	6.60
2	3.068	4.865	7.06	8.9	8.9	12.18
10	5.82	8.447	24.71	31.8	31.8	41.9
15	7.54	10.685	27.82	35.8	35.8	47.02
20	9.26	12.923	24.82	31.8	31.8	41.79
30	12.7	17.4	0.52	0	0	0.28
31	12.8	17.8	0.53	0	0	0.30
32	12.9	17.7	0.53	0	0	0.31

i: nombre d'échantillons appartenant à la deuxième classe
$M_1(i)$, $M_2(i)$: composantes de M(i)
$S_{11}(i)$, ..., $S_{22}(i)$: composantes de S(i)

tableau-1.1 évolution de la fenêtre entre 2 classes

Selon ce tableau, la relation entre les paramètres et la position de la fenêtre est illustrée par les graphes ci-dessous.

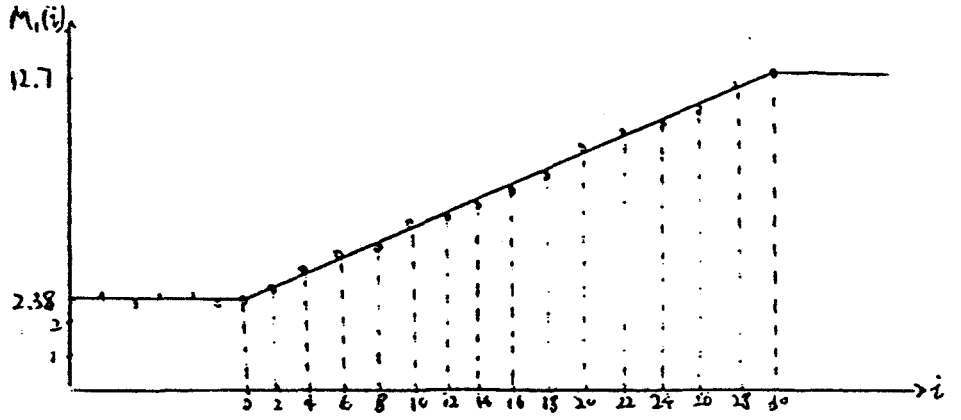


fig-1.16 relation entre i et $M_1(i)$

Les expériences sont conformes aux formules (1.2.34) et (1.2.35). Si la fenêtre se trouve au milieu des 2 classes, chaque élément de la matrice de covariance devient maximal. Le tableau suivant montre un autre exemple illustrant l'algorithme rapide et sa modification.

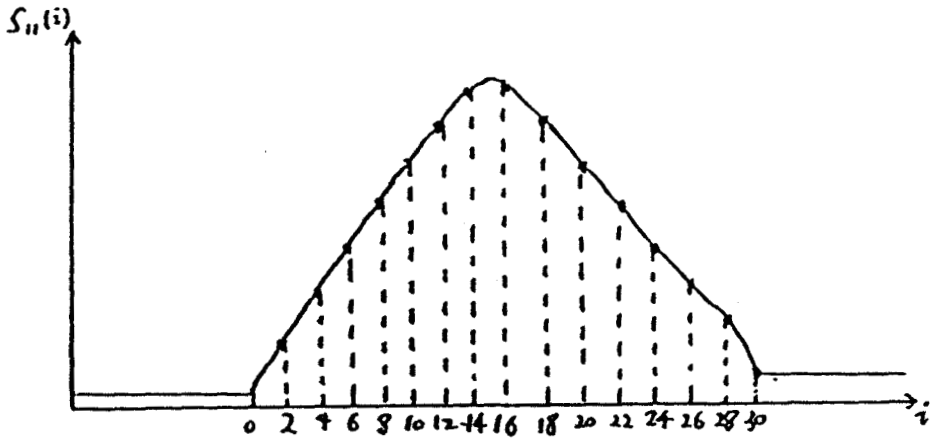


fig-1.17 relation entre i et $S_{11}(i)$

Le tableau ci-dessous fait apparaître:

- n_1 : numéro de l'échantillon intégré
- n_2 : numéro de la classe contenant la tête de fenêtre (nouvelle classe)
- n_3 : nombre d'échantillons appartenant à la nouvelle classe
- λ: valeurs propres de la fenêtre
- U_i (i=1,2): vecteurs propres de la fenêtre
- la longueur de fenêtre choisie est n=15

n_1	n_2	n_3	λ	U_1	U_2
1	1	16	0.018	0.729	-0.683
			0.011	0.683	0.729
5	1	20	0.034	0.975	0.222
			0.016	-0.222	0.975
6	2	1	2.108	0.717	-0.697
			0.028	0.697	0.717
15	2	10	8.088	0.730	-0.683
			0.022	0.683	0.730
19	2	14	2.068	0.710	-0.704
			0.01	0.704	0.710
20	2	15	0.01	0.996	0.086
			0.016	-0.086	0.996
36	3	1	0.024	0.718	0.695
			10.37	-0.695	0.718
76	1	1	0.05	0.766	0.643
			2.970	-0.643	0.766
91	1	16	0.018	0.730	-0.683
			0.011	0.683	0.730

tableau-1.2 éléments propres de la fenêtre en évolution parmi 3 classes (d=2 et n=15)

Selon le tableau-1.2, les conclusions du chapitre précédent

sont vérifiées par la simulation. Après évolution des échantillons parmi plusieurs classes, le retour à l'état initial est toujours assuré.

II.5.3 TEMPS DE CALCUL

Le tableau-1.3 donne les temps de calcul en centième de secondes. Les calculs ont été effectués sur IBM-PC/AT avec un processeur 80286 (quartz=8MHz). Les programmes ont été écrits en TURBO-PASCAL.

i(numéro d'itération intégré)	1	10	20	30	40	50	60	70	80
temps (d=2)	17	39	44	16	11	17	11	16	39
temps (d=3)	44	33	54	39	72	33	27	33	38

tableau-1.3 évaluation du temps de calcul

Evidemment, le temps de calcul est différent à chaque glissement de fenêtre. Le temps le plus court correspond au cas où les d racines différentes de l'équation (1.2.15) sont résolues directement par l'algorithme rapide. Le temps le plus long correspond au cas où aucune racine ne peut être résolue par l'algorithme rapide. Dans ce cas, d racines de (1.2.15) se calculent à partir de l'algorithme A_2. Dans les autres cas, une partie des racines de (1.2.15) est obtenue par l'algorithme rapide dans chaque boucle d'exécution. Après un certain nombre de boucles, on obtient l'ensemble des solutions de (1.2.15).

Dans [LAKEL, 1987] et [VASSEUR, 1990], une comparaison de temps de calcul a été effectuée entre l'algorithme rapide et la méthode de diagonalisation. Cette comparaison montre de meilleures performances de l'algorithme rapide.

II.6 CONCLUSION

L'objectif de ce chapitre était de mettre en évidence la détection et la caractérisation des processus dynamiques en temps réel. En réalité, les processus dynamiques sont parfois difficiles à modéliser et les paramètres caractéristiques ne sont pas toujours accessibles. Dans ce cas, l'observation des échantillons temporels de la sortie apparaît alors comme une ressource essentielle permettant l'analyse des processus.

Dans ce chapitre, l'évolution temporelle du processus a été envisagée par une fenêtre glissante dans le temps comprenant un nombre fixé d'échantillons d'observation du processus. Cette fenêtre peut être utilisée pour caractériser l'évolution des paramètres caractéristiques sur la distribution des échantillons dans le temps.

L'observation du glissement de fenêtre permet d'obtenir une image immédiate et précise sur la situation actuelle du processus. Selon cette observation, les caractéristiques actuelles du processus (vecteur moyenne, valeurs propres et vecteurs propres) sont estimées par l'application des algorithmes présentés ci-dessus. Ces algorithmes peuvent être mis en parallèle et être exécutés sur un système multi-processeurs.

La précision d'estimation de ces algorithmes est basée sur les échantillons observés. Si les échantillons sont fortement perturbés, les valeurs estimées ne peuvent pas refléter la situation actuelle du processus. Dans le chapitre suivant, on propose un algorithme d'estimation des paramètres à partir d'échantillons fortement perturbés. Cet algorithme peut effectivement éliminer les perturbations ramenées aux échantillons observés et permettre d'obtenir les valeurs estimées correctes.

CHAPITRE III

ESTIMATION DES PARAMETRES STATISTIQUES A PARTIR D'ECHANTILLONS FORTEMENT PERTURBES

III.1 METHODE DU MAXIMUM DE VRAISEMBLANCE MODIFIEE

III.1.1 MESURE AVEC DE FORTES PERTURBATIONS

Lors de la mesure des paramètres caractéristiques d'une population, il apparaît parfois de très fortes perturbations dues à la technologie et/ou à l'environnement. Il est souvent difficile de détecter les erreurs provoquées par ces perturbations. Il est donc important d'élaborer une méthode d'estimation des valeurs correctes des éléments caractéristiques du processus dans des conditions de fortes perturbations sur les échantillons de la population observée.

III.1.2 METHODE ADOPTEE

La méthode du maximum de vraisemblance, qui est fréquemment utilisée, offre de bonnes performances asympotiques. L'algorithme récursif rapide présenté dans le Chapitre II est également basé sur cette méthode. Toutefois, il est très sensible aux fortes perturbations. La méthode du maximum de vraisemblance modifiée (MML), proposée par [TIKU, 1967], est basée sur la mise en ordre des échantillons observés pour l'estimation des paramètres. Cette approche se montre plus performante que la méthode du maximum de vraisemblance dans la convergence et la robustesse, tout en conservant les performances asympotiques de la méthode précédente. En cas de forte perturbation imposée aux échantillons observés, on peut, par cette méthode, éliminer l'influence des bruits et obtenir une estimation correcte des paramètres. Dans la commande d'un système, il est également important de développer un schéma d'identification insensible aux variations extérieures tout en conservant de bonnes qualités d'estimation des paramètres du système [ASTRÖM, 1980]. Dans ces conditions, l'identification des paramètres du système peut s'effectuer à partir de données

très perturbées en appliquant la MML [PUTHENPURA et SINHA, 1985].

Les propriétés importantes de la MML ont été discutées dans [TIKU, 1980] et [TIKU et SINGH, 1982]. Selon ces références, la MML est considérablement plus efficace que d'autres estimations. Elle peut être appliquée dans des cas différents où les fonctions de densité de probabilité sont usuelles: Gaussienne, gamma, Cauchy, etc.

III.1.3 APPLICATION DE LA MML DANS L'ESPACE MULTI-DIMENSIONNEL

Les échantillons pris en compte dans la méthode ci-dessus sont tous unidimensionnels. Pour les systèmes étudiés dans ce mémoire, l'espace d'observation est multi-dimensionnel. Il est donc nécessaire de reconstruire l'algorithme d'estimation en tenant compte de nouvelles contraintes, [ZENG et VASSEUR, 1991].

Dans ce chapitre, on propose un algorithme d'estimation des paramètres statistiques d'une population à partir d'échantillons stochastiques. Ces échantillons sont fortement perturbés dans le temps et l'objectif de l'algorithme est d'estimer les caractéristiques de la population (vecteur moyenne et matrice de covariance de la distribution de la population). Cette estimation se réalise dans un espace multi-dimensionnel en utilisant la méthode du maximum de vraisemblance.

III.2 ESTIMATION DES ELEMENTS CARACTERISTIQUES

III.2.1 CLASSEMENT DES ECHANTILLONS OBSERVES

Soit n échantillons à d dimensions recueillis lors de l'observation d'une population: Y_1 , Y_2, ..., Y_n, où Y_i (i=1, 2, ..., n) est un vecteur dans l'espace à d dimensions.

Les échantillons non-perturbés respectent la loi normale de distribution dont la fonction de densité a été précisé dans (1.1.1). La méthode exposée ci-dessous permet l'estimation de M et S.

Par ailleurs, on fait l'hypothèse que parmi ces n échantillons le nombre des échantillons fortement perturbés r est petit. Dans la pratique, la méthode est efficace tant que la proportion r/n est inférieure à 25%.

Dans un premier temps, pour déterminer les r échantillons les plus perturbés, il faut ordonner les n vecteurs Y_i selon leur degré croissant de perturbation.

La moyenne des $\{Y_i\}$ (i=1, 2, ..., n) est notée par

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i \qquad (1.3.1)$$

On peut facilement calculer les normes des vecteurs $\{Y_i - \bar{Y}\}$:

$$\text{norm}_i = \left\| Y_i - \bar{Y} \right\| = \left[(Y_i - \bar{Y})^T (Y_i - \bar{Y}) \right]^{1/2} \qquad (1.3.2)$$

Les échantillons peuvent alors être ordonnés selon l'ordre croissant de ces normes.

On obtient donc les conditions nécessaires et suffisantes suivantes:

$$i \leq j \iff Y_i \propto Y_j \iff \text{norm}_i \leq \text{norm}_j \qquad (1.3.3)$$

où "$\propto$" signifie "moins perturbé que"

Dans ces conditions, Y_{n-r+1}, ..., Y_{n-1}, Y_n peuvent être considérés comme les r échantillons les plus perturbés au cours

de l'observation. Les échantillons normaux sont donc les suivants:

$$Y_1, Y_2, \ldots, Y_{n-r} \tag{1.3.4}$$

III.2.2 ESTIMATION DE M ET S

Selon l'idée de la MML, la fonction de vraisemblance peut être écrite sous la forme suivante:

$$\begin{aligned} L&=f(Y_1, Y_2, \ldots, Y_{n-r}|M,S) \\ &=\{P(\|Y-\bar{Y}\|\geq\|Y_{n-r}-\bar{Y}\|\}^r\prod_{i=1}^{n-r} f(Y_i|M,S) \\ &=\{P(\|Y-\bar{Y}\|\geq\|Y_{n-r}-\bar{Y}\|\}^r\prod_{i=1}^{n-r}\frac{1}{(2\pi)^{d/2}|S|^{1/2}}\exp(-\frac{1}{2}(Y_i-M)^T S^{-1}(Y_i-M)) \end{aligned} \tag{1.3.5}$$

Les équations pour l'estimation de M et S sont:

$$\begin{cases} \dfrac{\partial L}{\partial m_i} = 0 \qquad \forall i\in\{1,\ldots,d\} & (1.3.6) \\ \dfrac{\partial L}{\partial s_{ij}} = 0 \qquad \forall i,j\in\{1,\ldots,d\} & (1.3.7) \end{cases}$$

En notant $R = \|Y_{n-r}-\bar{Y}\|$, on obtient $P(\|Y-\bar{Y}\|\geq R) = \int_D f(Y)dY$, où $f(Y)$ est la fonction de densité de Y et D: $(Y-\bar{Y})^T(Y-\bar{Y})\geq R^2$

De plus, on a:

$$\frac{\partial f(Y)}{\partial M} = f(Y)S^{-1}(Y-M) \tag{1.3.8}$$

A partir de (1.3.5) et (1.3.6), il vient:

$$\int_D f(Y)dY \cdot \sum_{i=1}^{n-r} S^{-1}(Y_i-M) + r\cdot S^{-1}\int_D (Y-M)f(Y)dY = 0 \tag{1.3.9}$$

S étant une matrice définie positive, on obtient:

$$M = \frac{\sum_{i=1}^{n-r} Y_i \int_D f(Y)dY + r\int_D Yf(Y)dY}{n\int_D f(Y)dY} \qquad (1.3.10)$$

Si on note $A=S^{-1}$, (1.3.7) peut être transformée en $\frac{\partial L}{\partial a_{ij}} = 0$, $\forall i,j\in\{1,\ldots,d\}$. En appliquant la méthode de [ANDERSON,1958], on obtient les équations suivantes:

$$\begin{cases} [(n-r)|A|^{-1}A_{ii} - \sum_{k=1}^{n-r} z_{ki}^2]\int_D f(Y)dY + r\frac{A_{ii}}{|A|}\int_D f(Y)dY - r\int_D z_i^2 f(Y)dY = 0 \\ [(n-r)|A|^{-1}A_{ij} - \sum_{k=1}^{n-r} z_{ki}z_{kj}]\int_D f(Y)dY + r\frac{A_{ij}}{|A|}\int_D f(Y)dY - r\int_D z_i z_j f(Y)dY = 0 \end{cases} \qquad (1.3.11)$$

où z_{ki} = $i^{ème}$ élément de $(Y_k - M)$
z_j = $j^{ème}$ élément de $(Y-M)$

Pour $\forall i,j\in\{1, 2, \ldots, d\}$, on peut déduire de (1.3.11) l'équation suivante:

$$s_{ij} = \frac{A_{ij}}{|A|} = \frac{\sum_{k=1}^{n-r} z_{ki}z_{kj}\int_D f(Y)dY + r\int_D z_i z_j f(Y)dY}{n\int_D f(Y)dY} \qquad (1.3.12)$$

où A_{ij} est un élement de la matrice adjointe de A

Il vient alors:

$$s_{ij} = \frac{1}{n}\sum_{k=1}^{n-r} z_{ki}z_{kj} + \frac{r}{n}\cdot\frac{\int_D z_i z_j f(Y)dY}{\int_D f(Y)dY} \qquad (1.3.13)$$

Selon (1.3.10) et (1.3.13), les équations pour l'estimation du vecteur moyenne et de la matrice de covariance sont donc données, après simplification, par:

$$\begin{cases} M = \frac{1}{n} \sum_{i=1}^{n-r} Y_i + \frac{r}{n} \frac{\int_D Yf(Y)dY}{\int_D f(Y)dY} \\ S = \frac{1}{n} \sum_{k=1}^{n-r} (Y_k - M)(Y_k - M)^T + \frac{r}{n} \frac{\int_D (Y-M)(Y-M)^T f(Y)dY}{\int_D f(Y)dY} \end{cases} \qquad (1.3.14)$$

Les équations (1.3.14) conduisent à des calculs difficiles pour la résolution de M et S. La section suivante fournit les approximations nécessaires au calcul des intégrales contenues dans (1.3.14) et l'algorithme récursif d'estimation de M et S.

III.3 ALGORITHME RECURSIF ET APPROXIMATION

III.3.1 APPROXIMATION DES EQUATIONS

Dans les équations (1.3.14), il est difficile de trouver un moyen pour calculer les intégrales sans approximation. En effet, chaque intégrale est exprimée comme fonction de M et S. Il faut donc transformer (1.3.14) en fonctions simples et calculables de M et S.

S peut se réécrire sous la forme suivante:

$$S = U\Lambda U^T \tag{1.3.15}$$

où U est la matrice orthogonale des vecteurs propres ($U^T U=I$) et Λ matrice diagonale des valeurs propres (λ_i) de S, soit: $\Lambda=\mathrm{diag}(\lambda_1, \ldots, \lambda_d)$.

En posant $T=U^T(Y-M)$, on obtient les conclusions suivantes:

$$\bar{T} = U^T(\bar{Y}-M) \tag{1.3.16}$$

$$D: \quad (T-\bar{T})^T(T-\bar{T}) \geq R^2 \tag{1.3.17}$$

$$\int_D Yf(Y)dY = M\int_D f(T)dT + U\int_D Tf(T)dT \tag{1.3.18}$$

Pour simplifier les calculs, il est nécessaire de transformer les intégrales du domaine D en D': $(T-\bar{T})^T(T-\bar{T}) \leq R^2$ ou $W^T W \leq R^2$ $(W=(w_1 \ w_2 \ldots \ w_d)^T=T-\bar{T})$.

Evidemment, on a:

$$\begin{cases} \int_D f(T)dT = 1 - \int_{D'} f(T)dT \\ \int_D Tf(T)dT = -\int_{D'} Tf(T)dT \\ \int_D TT^T f(T)dT = \Lambda - \int_{D'} TT^T f(T)dT \end{cases} \tag{1.3.19}$$

Dans l'hypothèse où la proportion d'échantillons fortement perturbée est limitée et où les échantillons non perturbés sont concentrés autour du vecteur moyenne, on peut considérer que le

domaine D' est petit.

Dans cette condition, f(T), qui est continue, peut être développée en série de Maclaurin au voisinage de $\bar{T}$ dans D'. Ceci donne à l'ordre 2:

$$\int_{D'} f(T)\,dT=\int_{D'} [f(\bar{T})+(T-\bar{T})^T \frac{\partial f(T)}{\partial T}\Big|_{\bar{T}}+(T-\bar{T})^T \frac{\partial^2 f(T)}{\partial T^2}\Big|_{\bar{T}}(T-\bar{T})]\,dT \qquad (1.3.20)$$

D' étant un domaine hypersphérique symétrique, (1.3.20) peut se réécrire sous la forme suivante:

$$\int_{D'} f(T)\,dT = f(\bar{T})[vol(d,R) + \sum_{i=1}^{d} g_{ii}\int_{D'} w_1^2\,dW] \qquad (1.3.21)$$

où $vol(d,R)=\int_{D'} dW$: volume de l'hypersphère de rayon R dans un espace à d dimensions et $G=\{g_{ij}\}$ avec $g_{ii}=\frac{2}{\lambda_i}+\frac{4\bar{t}_i^2}{\lambda_i^2}$,

$g_{ij}=\frac{4\,\bar{t}_i\,\bar{t}_j}{\lambda_i\lambda_j}$ $(i\neq j)$ pour $\forall i,j\in\{1,2,\ldots,d\}$.

De la même manière, on obtient l'approximation des 2 autres intégrales:

$$\int_{D'} Tf(T)\,dT=f(\bar{T})[\bar{T}(vol(d,R)+\sum_{i=1}^{d} g_{ii}\int_{D'} w_i^2\,dW)+V\int_{D'} w_1^2\,dW] \qquad (1.3.22)$$

où $V=(v_1\ v_2\ \ldots\ v_d)^T$ et $v_i=\frac{2\bar{t}_i}{\lambda_i}$ pour $\forall i\in\{1,\ldots,d\}$

$$\int_{D'} TT^T f(T)\,dT=f(\bar{T})[(vol(d,R)+\sum_{i=1}^{d} g_{ii}\int_{D'} w_1^2\,dW)\bar{T}\bar{T}^T+(\bar{T}V^T+V\bar{T}^T)\int_{D'} w_1^2\,dW$$

$$+(I\int_{D'} w_1^2\,dW+G_d\int_{D'} w_1^4\,dW)+G_l\int_{D'} w_1^2 w_2^2\,dW] \qquad (1.3.23)$$

$$\text{où}\ \begin{cases}(G_l)_{ij}=2g_{ij} & (i\neq j)\\ (G_l)_{ii}=\sum\limits_{\substack{k=1\\k\neq i}}^{d} g_{kk}\end{cases} \qquad G_d=\mathrm{diag}(g_{11},\ldots,g_{dd})$$

D'après les équations ci-dessus, on obtient des formules calculables d'estimation de M et S de la façon suivante:

$$\begin{cases} M=\dfrac{1}{n-r}\sum\limits_{i=1}^{n-r} Y_i-\dfrac{r}{n-r}Uf(\bar{T})\dfrac{[\mathrm{vol}(d,R)+\sum\limits_{i=1}^{d} g_{ii}\int_{D'} w_1^2\,dW]\bar{T}+V\int_{D'} w_1^2\,dW}{1-f(\bar{T})[\mathrm{vol}(d,R)+\sum\limits_{i=1}^{d} g_{ii}\int_{D'} w_1^2\,dW]} \\ S=\dfrac{1-\int_{D'} f(T)\,dT}{(n-r)-n\int_{D'} f(T)\,dT}\sum\limits_{k=1}^{n-r}(Y_k-M)(Y_k-M)^T-\dfrac{r}{(n-r)-n\int_{D'} f(T)\,dT}U\int_{D'} TT^T f(T)\,dT\cdot U^T \end{cases} \tag{1.3.24}$$

Les approximations des intégrales dans (1.3.24) sont données ci-dessous:

$$\begin{cases} \int_{D'} w_1^2\,dW=2\sum\limits_{i=1}^{p} u_i^2(\Delta w_1)\,\mathrm{vol}(d-1,\sqrt{R^2-u_i^2}) \\ \int_{D'} w_1^4\,dW=2\sum\limits_{i=1}^{p} u_i^4(\Delta w_1)\,\mathrm{vol}(d-1,\sqrt{R^2-u_i^2}) \\ \int_{D'} w_1^2 w_2^2\,dW=4\sum\limits_{i=1}^{p}\sum\limits_{j=1}^{q(i)} u_i^2 u_j^2(\Delta w_1)^2\,\mathrm{vol}(d-2,\sqrt{R^2-u_i^2-u_j^2}) \end{cases} \tag{1.3.25}$$

où $u_i=(i-0.5)(\Delta w_1)$ et p nombre de pas de discrétisation ($\Delta w_1=\frac{R}{p}$), q(i) est un entier maximal satisfaisant $u_i^2+u_{q(i)}^2\leq R^2$.

Par ailleurs, vol(d,R) a pour expression:

$$\begin{cases} vol(d,R) = \dfrac{2(2\pi)^m R^{2m+1}}{(2m+1)!!} & (d=2m+1) \\ vol(d,R) = \dfrac{(2\pi)^m R^{2m}}{(2m)!!} & (d=2m) \end{cases} \qquad (1.3.26)$$

Le calcul de (1.3.25) et (1.3.26) peut être effectué avant l'exécution de l'algorithme récursif.

III.3.2 ALGORITHME RECURSIF

Formellement, l'estimation de M et S se ramène donc à la résolution des équations (1.3.24). La résolution directe de ces équations non-linéaires est un problème délicat. On essaie donc d'élaborer un algorithme récursif de (1.3.24) pour l'obtention de M et S.

Les formules (1.3.24) peuvent se réécrire sous la forme suivante:

$$\begin{cases} M = f_1(M,S) \\ S = f_2(M,S) \end{cases} \qquad (1.3.27)$$

où f_1, f_2 sont des fonctions continues calculables.

Pour estimer S et M, on peut énoncer l'algorithme suivant:

Etape 0:

Les premières estimations M_0, S_0 pour M et S sont données par:

$$\begin{cases} M_0 = \dfrac{1}{n-r}\sum_{i=1}^{n-r} Y_i \\ S_0 = \dfrac{1}{n-r}\sum_{i=1}^{n-r} (Y_i - M_0)(Y_i - M_0)^T \end{cases} \qquad (1.3.28)$$

Dans le cas général, M_0 et S_0 sont proches de M et S.

Etape 1:

On effectue le calcul $f_1(M_0, S_0)$ et $f_2(M_0, S_0)$ selon les

équations (1.3.27). Ensuite, on détecte si $\|M_0 - f_1(M_0, S_0)\| \leq \varepsilon$ et $\|S_0 - f_2(M_0, S_0)\| \leq \varepsilon$, où $\|S - f_2(M,S)\| = \left[\sum_i \sum_j (S(i,j) - f_2(M,S)(i,j))^2\right]^{1/2}$ (ε est un réel fixé). Si ces conditions sont satisfaites, l'algorithme se termine; sinon, aller à l'étape 2.

Etape 2:

D'après la méthode itérative simple, on obtient les nouvelles valeurs de M_1 et S_1 de la façon suivante:

$$\begin{cases} M_1 = f_1(M_0, S_0) \\ S_1 = f_2(M_0, S_0) \end{cases} \tag{1.3.29}$$

Etape 3:

Transférer les valeurs de M_1 et S_1 aux M_0 et S_0. Retour à l'étape 1.

Si les conditions de l'étape 1 sont toutes vérifiées, les derniers M_0 et S_0 sont considérés comme solutions des équations (1.3.24), soit l'estimation du vecteur moyenne et de la matrice de covariance.

Nous discuterons ci-dessus la convergence de cet algorithme:

En effet, $M=f_1(M,S)$, $S=f_2(M,S)$ représentent respectivement les d et d×d équations non-linéaires. Ces K=d+d×d équations peuvent se réécrire sous la forme suivante:

$$X = g(X) \tag{1.3.30}$$

où $X=(m_1\ m_2\ \ldots\ m_d\ s_{11}\ \ldots\ s_{1d}\ \ldots\ s_{dd})^T$

et g(X) est le vecteur composé des éléments de f_1 et f_2 qui correspondent aux composants de X.

Evidemment, la solution de (1.3.30) est le résultat de l'estimation de M et S.

Selon les équations (1.3.29), les itérations effectuées dans cet algorithme suivent la formule suivante:

$$X^{N+1} = g(X^N) \tag{1.3.31}$$

D'après [BAKHVALOV, 1976] et [SIBONY et MARDON, 1982], cette méthode permet de résoudre X=g(X) si pour $\forall X_1, X_2 \in \Re^{d+d\times d}$, $\exists q$ $(0<q<1)$, l'application X=g(X) vérifie la condition:

$$\|g(X_1)-g(X_2)\| \leqslant q\|X_1-X_2\| \qquad (1.3.32)$$

Selon (1.3.28), la valeur initiale X^0 se trouve dans un petit voisinage de la solution **X**. L'ensemble des $X^0, X^1, \ldots, X^N, \ldots$, demeurent au voisinage restreint de la solution si l'algorithme est convergent. Ainsi, on peut approximer (1.3.31) de la manière suivante:

$$X^{N+1}-\mathbf{X} = g(X^N)-g(\mathbf{X}) \approx B(X^N-\mathbf{X}) \qquad (1.3.33)$$

$$\text{avec } B = \left[\frac{\partial g_i}{\partial x_j}\right]\Big|_x$$

A partir de (1.3.32) et (1.3.33), on obtient:

$$\|g(X_1)-g(X_2)\| \approx \left[(X_1-X_2)^T B^T B(X_1-X_2)\right]^{1/2} \qquad (1.3.34)$$

D'après la règle $B^T B = U_1^T \Lambda_1 U_1$ (où $\Lambda_1 = \mathrm{diag}(\lambda_{11}, \ldots, \lambda_{1K})$ et $U_1^T U_1 = I$), (1.3.33) peut se réécrire sous la forme suivante:

$$\|g(X_1)-g(X_2)\| = \left[E^T \Lambda_1 E\right]^{1/2} \qquad (1.3.35)$$

où $E = (e_1\ e_2\ \ldots\ e_K)^T = U_1(X_1-X_2)$

Par conséquent, la condition (1.3.32) est transformée en (1.3.36):

$$\sum_{i=1}^{K} \lambda_{1i} e_i^2 \leqslant q\cdot\sum_{i=1}^{K} e_i^2 \qquad (1.3.36)$$

X_1, X_2 étant deux vecteurs quelconques du voisinage de **X**, nous pouvons donner la condition de convergence de (1.3.30) de la manière suivante:

Pour $\forall i\in\{1,\ldots,K\}$, si $\lambda_{1i}<1$, on peut sélectionner $q=\max_i\{\lambda_{1i}\}$ afin que (1.3.36) soit satisfaite. Dans cette condition, l'algorithme est convergent.

Il est difficile de calculer la matrice B à partir des équations f_1 et f_2. Ainsi, les valeurs propres de $B^T B$ restent inconnues et on n'arrive pas à prouver théoriquement la

convergence de l'algorithme.

En pratique, tous les exemples de simulation montrent une bonne convergence de cet algorithme. Nous considérons donc que cette condition de convergence est automatiquement satisfaite par les équations spécifiques f_1 et f_2.

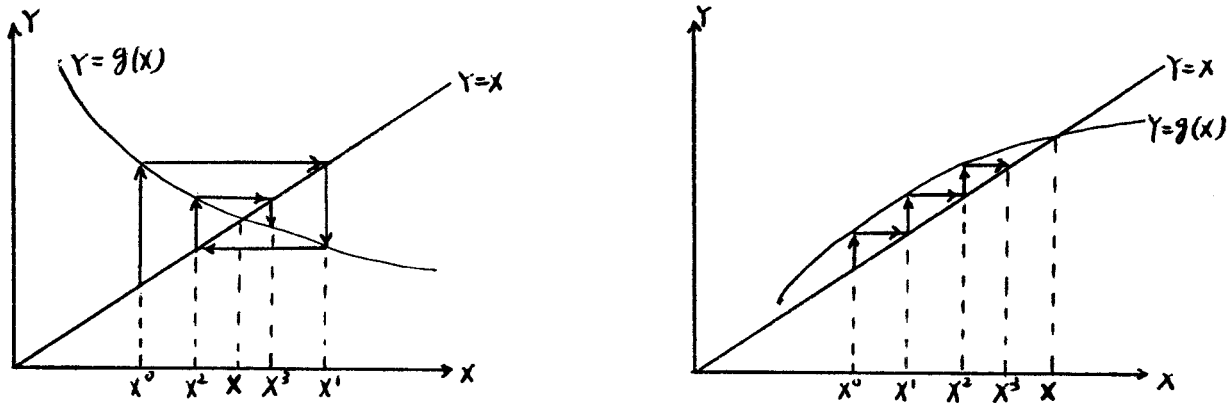


fig-1.18 trajectoire de convergence de l'algorithme (projection dans un plan)

III.4 RESULTATS DE SIMULATION ET REMARQUES FINALES

III.4.1 PRINCIPE DE SIMULATION

L'objectif de la simulation est de tester la performance de l'algorithme proposé et de comparer les paramètres estimés en appliquant les différents algorithmes (ML et MML). La procédure de la simulation est donnée ci-dessous:

Etape 1: Génération d'un ensemble d'échantillons non perturbés E et d'un ensemble d'échantillons perturbés E'.

Etape 2: Extraction d'un sous ensemble de n échantillons avec perturbation à partir de E et E':

Prendre n-r échantillons de E et estimer les M et S à partir de ces échantillons non perturbés (ALGO III). Prendre r échantillons de E' et mélanger ces r échantillons avec les n-r échantillons de E.

Etape 3: Estimer M et S à partir du mélange en appliquant l'algorithme MML (ALGO I) et l'algorithme ML (ALGO II).

Etape 4: Présenter les résultats dans un tableau et comparer ALGO I, ALGO II et ALGO III.

III.4.2 RESULTATS DE SIMULATION

Toutes les simulations sont effectuées avec $\varepsilon=10^{-6}$ et d=2, 3 et 4. Les résultats sont sonsignés dans les tableaux ci-dessous pour 4 exemples. Ces tableaux comparent les estimations des paramètres statistiques obtenues, grâce à l'algorithme MML (ALGO I), aux paramètres estimés sans considération des perturbations ramenées aux échantillons (ALGO II: algorithme ML), et aux paramètres estimés à partir des échantillons non perturbés (ALGO III).

d=2 n=20 r=5	vecteur moyenne	valeurs propres	vecteur propres
ALGO I	1.3625	0.13369	[0.8780 -0.47874]
	2.0330	0.26889	[0.47874 0.8780]
ALGO II	1.4007	0.21063	[0.9272 -0.4208]
	1.8915	0.46729	[0.4208 0.9072]
ALGO III	1.3592	0.1187	[0.8525 -0.5210]
	2.0563	0.2346	[0.5210 0.8525]

Tableau-1.4: simulation en dimension 2

d=2 n=50 r=10	vecteur moyenne	valeurs propres	vecteurs propres
ALGO I	1.339	0.963	[0.9388 0.3446]
	4.043	0.847	[-0.3446 0.9388]
ALGO II	1.413	2.4376	[0.9860 0.1665]
	3.961	1.0595	[-0.1665 0.9860]
ALGO III	1.311	0.9124	[0.9053 0.4175]
	4.097	0.8166	[-0.4175 0.9053]

Tableau-1.5: simulation en dimension 2

d=3 n=40 r=10	vecteur moyenne	valeurs propres	vecteurs propres
ALGO I	1.4389 2.5282 2.9670	0.0808 0.1896 0.1693	[0.9871 0.0029 -0.1600] [-0.0197 0.9945 -0.1032] [0.1588 0.1050 0.9812]
ALGO II	1.4442 2.4686 2.8554	0.0511 0.2382 0.3850	[0.9929 -0.1187 -0.0031] [0.1186 0.9906 0.0683] [-0.005 -0.0068 0.9977]
ALGO III	1.4360 2.5443 3.0293	0.0917 0.1994 0.1562	[0.9815 0.0123 -0.1547] [-0.0243 0.986 -0.0153] [0.2101 0.1217 0.9706]

Tableau-1.6: simulation en dimension 3

d=4 n=25 r=5	vecteur moyenne	valeurs propres	vecteurs propres
ALGO I	0.9680 1.9466 3.0039 4.1470	0.0745 0.1502 0.2330 0.2950	[0.9465 -0.2313 -0.2108 -0.0791] [0.0216 0.7405 -0.6094 -0.2824] [0.3091 0.6144 0.5255 0.5008] [-0.0906 -0.1435 -0.555 0.8144]
ALGO II	0.9417 1.8975 2.9252 4.0338	0.0376 0.1577 0.3512 0.5300	[0.9610 -0.27 -0.06 0.0064] [0.2513 0.9248 -0.1622 -0.235] [0.0927 0.1087 0.9785 -0.1485] [0.0694 0.2449 0.112 0.9606]
ALGO III	0.9715 1.9418 3.1024 4.0978	0.0638 0.1522 0.2160 0.2571	[0.9433 -0.2345 -0.2064 -0.9233] [0.0540 0.7069 -0.6255 -0.3006] [0.2165 0.6309 0.5067 0.5045] [-0.1024 -0.1533 -0.5744 0.7829]

Tableau-1.7: simulation en dimension 4

A partir de ces tableaux, il apparaît que les résultats de l'ALGO I et l'ALGO III sont assez proches et que la convergence de l'algorithme proposé ci-dessus est bonne. En comparaison de l'ALGO II, on obtient évidemment des valeurs propres plus petites. Après élimination des fortes perturbations, on obtient une distribution statistique plus concentrée dont les paramètres estimés caractérisent plus correctement la population observée.

Les temps de calcul et la récursivité peuvent être évalués de manière expérimentale.

Le tableau ci-dessous donne les temps de calcul en centièmes de secondes et le nombre d'itérations nécessaires à une exécution de l'algorithme récursif (ALGO I) sous la condition $\varepsilon=10^{-6}$ en fonction de la dimension de l'espace, de la population et du nombre d'échantillons.

d	2	2	3	4
n	20	50	40	25
t	132	181	379	708
b	18	21	19	18

t: temps de calcul (1/100 seconde)
b: nombre d'itérations

Tableau-1.8: temps de calcul et récursivité en fonction de d et n

Les temps de calcul de cet algorithme sont assez courts. Ils augmentent avec la dimension et avec le nombre d'échantillons. Le nombre d'itérations reste quasiment stable lors de l'augmentation de d.

III.4.3 REMARQUES FINALES

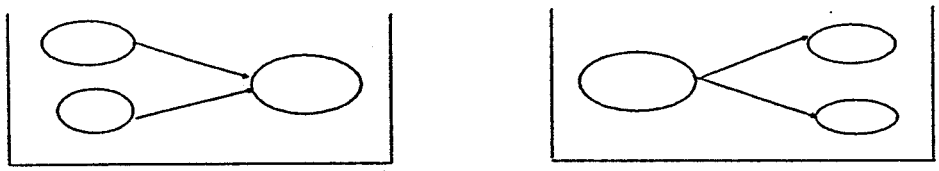
L'approche que nous avons presentée peut être appliquée pour conduire les caractéristiques externes d'un système par échantillonnage. Toutefois, au cours du fonctionnement, il apparaît que les caractéristiques des classes observées, évoluent de telle sorte que certaines classes peuvent être confondues ou, qu'au contraire, certaines autres classes ont tendance à l'éclatement. Ces phénomènes de fusion et de scission de classes sont étudiés au chapitre suivant.

CHAPITRE IV FUSION ET SCISSION DES CLASSES

IV.1 INTRODUCTION

IV.1.1 NOTIONS DE FUSION ET SCISSION

La fusion et la scission des données multi-dimensionnelles sont deux sujets importants de la reconnaissance des formes. L'objectif de cette approche est de regrouper dans l'espace multi-dimensionnel les modes ou classes dont les échantillons sont similaires (fusion), ou inversement de diviser un ensemble d'échantillons d'observation en différentes classes selon leurs degrés de dissemblance (scission).



fusion scission

fig-1.19 fusion et scission des classes

IV.1.2 METHODES GENERALES DE FUSION

Pour la fusion, il est important de définir des critères pour mesurer la séparabilité ou la ressemblance entre les classes à fusionner. La meilleure partition des classes correspond à une situation où le critère correspondant de fusion est le plus favorable. Les techniques utilisées pour la fusion peuvent être classifiées en 4 catégories de la manière suivante:

(a) méthode hiérarchique métrique:

Cette méthode permet d'établir une hiérarchie de partitions sous forme d'arbre de classification [CAILLIEZ et PAGE, 1976], [DUDES et JAIN, 1979], [GONDRAN, 1976].

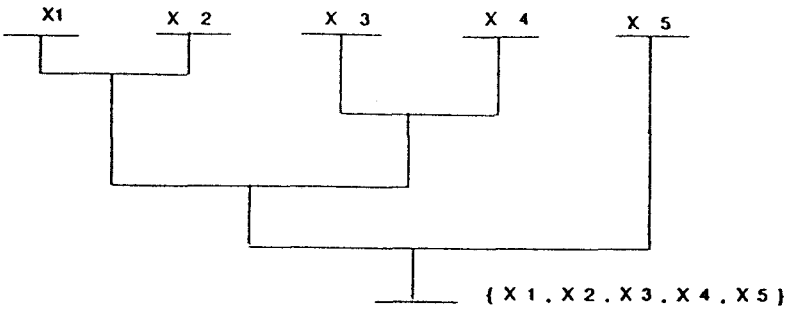


fig-1.20 approche hiérarchique

Dans cette procédure, on part des feuilles (échantillons individuels) pour aboutir à la racine (ensemble de tous les échantillons d'observation) en fusionnant deux à deux les classes similaires.

(b) méthode typologique:

Cette méthode permet de définir des sous ensembles d'objets (sous graphes complets symétriques) tels qu'à l'intérieur de ceux-ci, les objets soient proches ou semblables entre eux. La ressemblance ou la distance entre objets se calcule sur des graphes de connexion des objets [DEGOT et HUALDE, 1975].

(c) méthode informationnelle:

Cette méthode permet de mettre en évidence la notion d'information entre variables appelée transinformation [WATANABE, 1981]. La meilleure partition des classes est obtenue en minimisant la fonction d'entropie définie a priori.

(d) méthode basée sur la minimisation d'une fonction objective métrique:

Cette méthode est plus souvent utilisée que les autres. Parmi les applications de cette méthode, l'algorithme K-MEANS ou ISODATA [DIDAY, 1979] est le plus connu. On peut répétitivement exécuter l'algorithme K-MEANS à partir des différentes classes initiales pour sélectionner une partition qui correspond à la plus petite valeur de la fonction associée au critère. Pour l'obtention d'une solution efficace de ce problème, de nombreuses techniques sont utilisées pour modifier la stratégie K-MEANS [KITTLER, 1988], [ISMAIL et KAMEL, 1989].

Dans la plupart des méthodes présentées ci-dessus, le nombre de classes de la meilleure partition est connu. Si ce nombre reste inconnu, il est necéssaire d'en effectuer une estimation. [DUBES, 1987] et [DAVIES et BOULDIN, 1979] définissent des

indices internes pour trouver le meilleur nombre de classes. Ce nombre correspond à une partition où l'indice est minimal.

IV.1.3 PRINCIPE DE LA METHODE DE FUSION PROPOSEE

Ce chapitre fournit une nouvelle méthode de fusion, basée sur l'évolution d'un indice interne pour tester la meilleure partition.

D'après les analyses des chapitres précédents, les données (échantillons d'observation) respectent les lois de distribution normale. Ainsi, la forme de chaque classe peut être représentée par une hyper-ellipse [POSTAIRE, 1987].

Pour simplifier la procédure de fusion, nous faisons les hypothèses suivantes:

- Chacune des classes initiales contient un nombre suffisant d'échantillons pour estimer les paramètres statistiques (le vecteur moyenne et la matrice de covariance). Sous cette hypothèse, le cas des échantillons isolés n'est pas envisagé.

- Après fusion de deux classes dans l'espace euclidien multi-dimensionnel, les échantillons de la nouvelle classe respectent également les lois de distribution normale.

- Avant de lancer la procédure de fusion, on définit la constante k_{max}, comme le nombre de classes dans la partition initiale. La meilleure partition pour tous les échantillons se trouve dans l'un des états rencontrés au cours de la procédure de fusion. Cet état correspond à la valeur minimale de l'indice interne (cf fig-1.21).

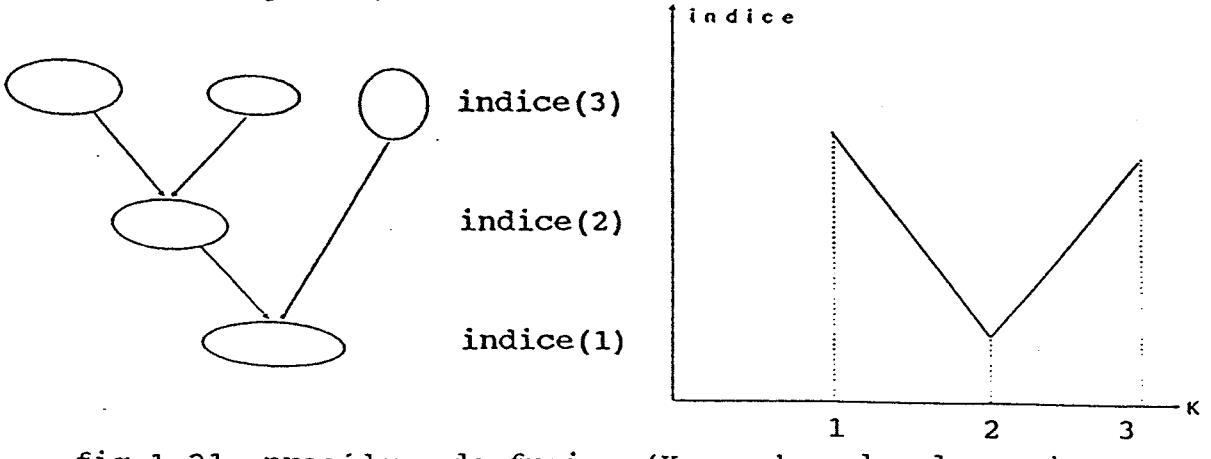


fig-1.21 procédure de fusion (K: nombre de classes)

La fig-1.21 montre un exemple de l'algorithme de fusion présenté dans ce chapitre. Dans cet exemple, le nombre de

classes de la meilleure partition est 2.

En général, une bonne partition de classes des échantillons doit satisfaire les conditions suivantes:

- bonne séparabilité entre des différentes classes:

Dans une telle partition, chaque classe doit montrer des caractéristiques différentes des autres classes. Dans l'espace multi-dimensionnel, ces caractéristiques traduisent l'isolement de chaque classe par rapport aux autres.

- bonne compacité au sein de chaque classe:

Les échantillons d'observation dans une même classe doivent se concentrer sur leur vecteur moyenne. S'il y a trop d'espace vide dans l'hyper-ellipse de cette classe ou s'il y a plusieurs agglomérations d'échantillons à l'intérieur de cette classe, la compacité devient mauvaise.

Dans la Section IV.2, on définit le degré de séparabilité et le degré de compacité pour notre problème spécifique. On propose ensuite un algorithme pour trouver la meilleure partition par minimisation de l'indice de fusion. Cet indice est défini comme une combinaison linéaire du degré de séparabilité et du degré de compacité. Dans la Section IV.3, trois exemples de simulation sont donnés pour montrer les performances de l'algorithme.

IV.1.4 METHODES GENERALES DE SCISSION

La scission est une procédure inverse de la fusion. Par éclatement d'une classe en 2, la procédure de scission permet de trouver la meilleure partition caractérisant tous les échantillons d'observation.

On donne ci-dessous un résumé des méthodes de scission fréquemment utilisées:

a) méthode heuristique:

Cette méthode permet de trouver une liste de points critiques, qui sont supposés appartenir à la frontière entre deux classes éclatées. Le contour de la classe initiale est défini a priori par l'ensemble des échantillons d'observation. Ensuite, en minimisant une fonction de coût, on trace une trajectoire de frontière qui connecte tous ces points critiques [LESTER, WILLIAMS, WEINTRAUB et BRENNER, 1979].

b) méthode utilisant la théorie des ensembles flous:

Une relation floue est définie sur les points critiques. L'éclatement d'une classe se fait par optimisation du critère caractérisant ces relations floues bien définies [VANDERHEYDT, OOSTERLINCK et BERGHE, 1981].

c) méthode d'analyse de courbure:

Pour une classe dont le contour est bien défini, l'analyse de courbure est souvent utilisée pour décrire la forme de ce contour ou pour en détecter les points critiques. La trajectoire d'éclatement de classe est tracée en calculant la courbure sur ces points critiques [LIANG, 1989].

IV.1.5 PRINCIPE DE LA METHODE DE SCISSION PROPOSEE

Dans ce chapitre, on effectue la scission d'une classe en définissant un indice interne et une fenêtre d'observation.

Pour diviser les échantillons d'une classe en 2, on prend un sous ensemble (fenêtre) d'échantillons dans cette classe. Cette fenêtre évolue selon une opération d'ajout de nouveaux échantillons et de retrait d'anciens échantillons (cf l'algorithme du Chapitre II). Les appartenances des échantillons à la classe sont décidées d'après les effets de cette opération. De cette manière, les échantillons sont séparés en 2 nouvelles classes. Chacune de ces nouvelles classes peut elle même poursuivre la procédure d'éclatement jusqu'à ce que le nombre de classes soit égal à k_{max} défini a priori. A partir de l'état k_{max}, on exécute la procédure de fusion définie ci-dessus afin de trouver la meilleure partition correspondant à l'indice interne minimal. Par conséquent, la recherche de la meilleure partition par scission s'effectue à travers une procédure d'éclatement suivie d'une procédure de fusion.

Dans la Section IV.4, on présente l'algorithme de scission en détail. La Section IV.5 fournit les résultats de simulation pour quelques exemples de scission.

IV.2 ALGORITHME DE FUSION

IV.2.1 DEFINITION DE LA SEPARABILITE GENERALE

Supposons qu'il existe K classes à fusionner: $C_1, C_2, \ldots, C_K$ $(1 \leqslant K \leqslant k_{max})$. On définit l'indice interne I(K) selon le degré de compacité au sein de chaque classe et le degré de séparabilité entre ces classes.

On commence par définir le degré de ressemblance R_{ik} entre 2 classes quelconques C_i et C_k $(i,k \in \{1,2,\ldots,K\})$.

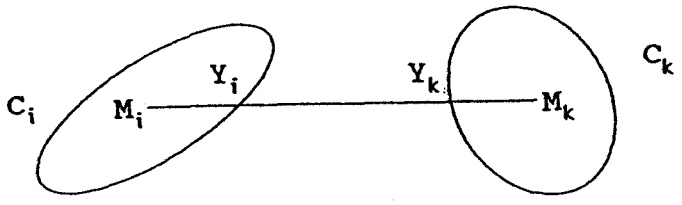


fig-1.22 Ressemblance entre deux classes C_i et C_k

Les C_i et C_k sont respectivement caractérisées par les les paramètres statistiques M_i, S_i et M_k, S_k où M_i, M_k sont vecteurs moyennes de dimension d et S_i, S_k matrices de covariance de dimension d×d. Les contours de C_i et C_k sont représentés par les équations suivantes:

$$\begin{cases} (Y-M_i)^T S_i^{-1} (Y-M_i)=1 & \text{pour le contour de } C_i \quad (1.4.1) \\ (Y-M_k)^T S_k^{-1} (Y-M_k)=1 & \text{pour le contour de } C_k \quad (1.4.2) \end{cases}$$

En traçant la droite $M_i M_k$ (fig-1.22), nous obtenons 2 points d'intersection avec les contours de C_i et C_k, notés Y_i et Y_k.

Le degré de ressemblance R_{ik} est alors défini de la façon suivante:

$$R_{ik} = \frac{\|M_i - Y_i\| + \|M_k - Y_k\|}{\|M_i - M_k\|} \qquad (1.4.3)$$

où $\|M_i - M_k\|$ est la distance euclidienne entre M_i et M_k,

soit: $\|M_i - M_k\| = [(M_i - M_k)^T (M_i - M_k)]^{1/2}$ (1.4.4)

Y_i et Y_k étant deux points de la droite $M_i M_k$, nous pouvons écrire:

$$Y_i - M_i = f_1 \cdot (M_k - M_i) \tag{1.4.5}$$

$$Y_k - M_k = f_2 \cdot (M_k - M_i) \tag{1.4.6}$$

où f_1 et f_2 sont deux nombres réels

Evidemment, Y_i, Y_k satisfont respectivement aux équations (1.4.1) et (1.4.2). En reportant (1.4.5) dans (1.4.1), on obtient:

$$f_1^2 (M_k - M_i)^T S_i^{-1} (M_k - M_i) = 1 \tag{1.4.7}$$

De la même manière, en reportant (1.4.6) dans (1.4.2), on obtient:

$$f_2^2 (M_k - M_i)^T S_k^{-1} (M_k - M_i) = 1 \tag{1.4.8}$$

d'où:

$$\begin{cases} |f_1| = \dfrac{1}{[(M_k - M_i)^T S_i^{-1} (M_k - M_i)]^{1/2}} \\ |f_2| = \dfrac{1}{[(M_k - M_i)^T S_k^{-1} (M_k - M_i)]^{1/2}} \end{cases} \tag{1.4.10}$$

Le degré de ressemblance peut alors se réécrire sous la forme suivante:

$$R_{ik} = |f_1| + |f_2| \tag{1.4.11}$$

Selon l'analyse du Chapitre II, les matrices de covariance peuvent se décomposer sous la forme suivante:

$$\begin{cases} S_i = U_i \Lambda_i U_i^T \\ S_k = U_k \Lambda_k U_k^T \end{cases} \tag{1.4.12}$$

où $\Lambda_i = \mathrm{diag}(\lambda_{i1}, \ldots, \lambda_{id})$, $\Lambda_k = \mathrm{diag}(\lambda_{k1}, \ldots, \lambda_{kd})$ et $U_i^T U_i = I$, $U_k^T U_k = I$.

$$\text{En notant} \begin{cases} U_i^T (M_k - M_i) = Z_{ik} \\ U_k^T (M_k - M_i) = Z_{ki} \end{cases} \tag{1.4.13}$$

Il vient:

$$\begin{cases} |f_1| = \dfrac{1}{[Z_{ik}^T \Lambda_i^{-1} Z_{ik}]^{1/2}} \\ |f_2| = \dfrac{1}{[Z_{ki}^T \Lambda_k^{-1} Z_{ki}]^{1/2}} \end{cases} \qquad (1.4.14)$$

On obtient:

$$R_{ik} = \frac{1}{[Z_{ik}^T \Lambda_i^{-1} Z_{ik}]^{1/2}} + \frac{1}{[Z_{ki}^T \Lambda_k^{-1} Z_{ki}]^{1/2}} \qquad (1.4.15)$$

Evidemment, plus R_{ik} est petit, meilleure est la séparabilité entre les classes C_i et C_k. Autrement dit, C_i et C_k sont inséparables si R_{ik} est trop grand.

On définit alors ci-dessous le degré de ressemblance de la classe C_k avec toutes les autres classes.

$$R_k = \underset{i \neq k}{\text{maximum}}\{R_{ik}\} \qquad (1.4.16)$$

R_k est le degré de séparabilité de C_k avec les autres classes. Plus R_k est petit, plus la séparabilité de C_k avec son environnement est bonne.

La séparabilité générale pour les K classes courantes S(K) peut être définie comme la moyenne de tous les R_k (k=1, 2, ..., K):

$$S(K) = (1/K)\sum_{k=1}^{K} R_k \text{ pour } K>1 \qquad (1.4.17)$$

Lorsque K=1, il n'existe qu'une classe et on définit S(1)=0.

IV.2.2 DEFINITION DE LA COMPACITE GENERALE

Dans une partition de K classes, la compacité d'une classe C_i, noté P_i ($\forall i \in \{1,2,\ldots,K\}$), est définie par la proportion du nombre d'échantillons à l'extérieur du contour de C_i sur le nombre total d'échantillons. Une bonne compacité se traduit donc par une faible valeur de P_i.

Soit Y un échantillon quelconque de C_i, la distance de MAHALOBIS entre Y et M_i est calculé par

$$d_i(Y) = (Y-M_i)^T S_i^{-1} (Y-M_i)$$

Selon l'équation (1.4.1), on obtient la conlusion suivante:

Si $d_i(Y)>1$, Y se trouve à l'extérieur du contour de C_i

Si $d_i(Y)<1$, Y se trouve à l'intérieur du contour de C_i

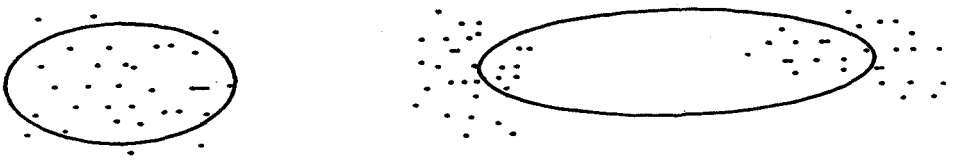
Ainsi, la définition de P_i peut s'écrire de la manière suivante:

$$P_i = \frac{\text{Card}\{Y \mid Y \in C_i \text{ et } d_i(Y)>1\}}{\text{Card}\{Y \mid Y \in C_i\}} \qquad (1.4.18)$$

La compacité générale est alors définie par

$$C(K) = \frac{1}{K}\sum_{i=1}^{K} P_i \quad (1 \leqslant K \leqslant k_{max}) \qquad (1.4.19)$$

Supposons qu'on obtienne une nouvelle classe après la fusion des classes C_j et C_k à l'étape K-1. Evidemment, plus les classes C_j et C_k sont proches, plus le contour de la nouvelle classe fusionnée englobe d'échantillons et plus la valeur de C(K) est faible. Ceci traduit une bonne compacité des échantillons à l'intérieur de la nouvelle classe (cf fig-1.23).



(a) bonne compacité (b) mauvaise compacité

fig-1.23 différents cas de compacité

Généralement, la compacité générale C(K) augmente au cours de la procédure de fusion deux à deux tandis que le nombre de classes K diminue. Autrement dit, plus on effectue de fusions pour des classes éloignées, moins les échantillons au sein des classes fusionnées sont compacts.

IV.2.3 ALGORITHME DE RECHERCHE DE LA MEILLEURE PARTITION PAR FUSION

On définit alors l'indice interne de partition de la manière suivante:

$$I(K) = S(K) + \beta C(K) \qquad (1.4.20)$$

où β est un nombre réel positif défini a priori.

La meilleure partition est obtenue en trouvant un I(K) minimal pour $K=k_{max},\ldots,2,1$. D'après les analyses précédentes, une telle partition possède une bonne séparabilité entre les classes et une bonne compacité à l'intérieur de chaque classe.

L'algorithme de fusion peut alors s'énoncer de la façon suivante:

ETAPE 1 (initialisation):

1.1 Choix des paramètres k_{max} et β.

1.2 Estimation des paramètres M_a, S_a et P_a pour a=1, 2, ..., k_{max}.

ETAPE 2: Evaluation des I(K) pour $K=k_{max}, \ldots, 1$.

Pour $K=k_{max}$ jusqu'à 1

Faire

2.1 Recherche des deux classes C_i et C_k telle que R_{ik} vérifie la condition ci-dessous:

$$R_{ik} = \underset{a\neq b}{\text{minimum}}\{R_{ab}\} \quad \forall a,b\in\{1,2,\ldots,K\} \qquad (1.4.21)$$

où les R_{ab} se calculent à partir de M_a, M_b et S_a, S_b.

2.2 Fusionner C_i et C_k et calculer l'indice interne I(K). Estimer le vecteur moyenne, la matrice de covariance et la compacité de la nouvelle classe.

Fin_Faire

Fin_Pour

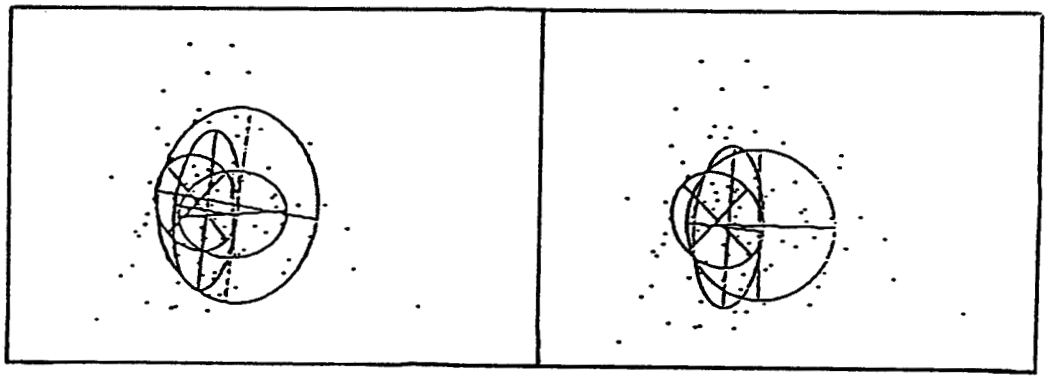
ETAPE 3: Examiner les valeur de I(K) pour $K=1,2,\ldots,k_{max}$. La meilleure partition correspond à celle où I(K) est minimal.

IV.3 RESULTATS DE SIMULATION DE L'ALGORITHME DE FUSION

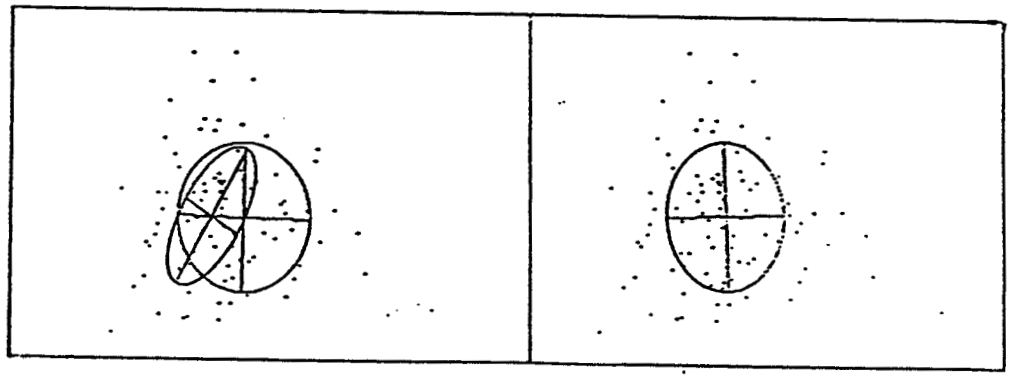
IV.3.1 RESULTATS DE SIMULATION

L'algorithme de fusion se déroule de $K=k_{max}$ à K=1 selon la procédure récursive présentée co-dessus. Dans chaque partition K, les deux classes les plus proches sont fusionnées pour former une nouvelle classe dans la partition K-1. On effectue la simulation sur divers distributions de données, avec d=2 et d=3. Les classes initiales de données sont générées de manière artificielle. Ensuite, on donne les résultats de simulation sur 3 exemples.

Exemple 1 (d=2, k_{max}=4, β=20):



(a) K=4 (b) K=3



(c) K=2 (d) K=1

fig-1.24 déroulement de la fusion pour l'exemple 1

K	4	3	2	1
S(K)	8.14	4.84	3.01	0
C(K)	0.589	0.550	0.581	0.592
I(K)	19.93	15.84	14.63	11.84

tableau 1.9 résultats numériques de fusion pour l'exemple 1

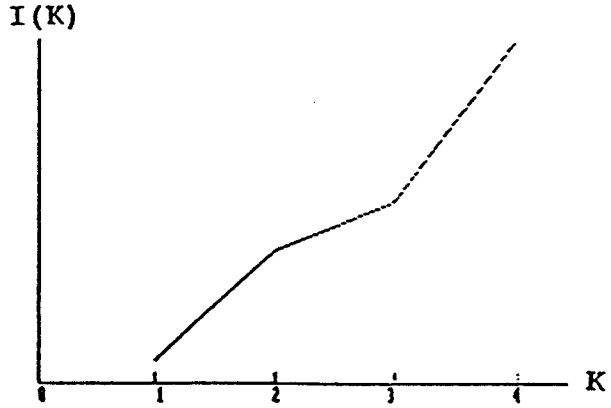


fig-1.25 évolution de l'indice au cours de la fusion (ex1)

Selon la fig-1.25, il apparaît que la meilleure partition correspond à K=1. Ceci se traduit parfaitement bien sur la fig-1.24 où, pour les états K=2, 3 et 4 les classes de la partition se superposent.

Exemple 2: (d=2, k_{max}=5, β=20):

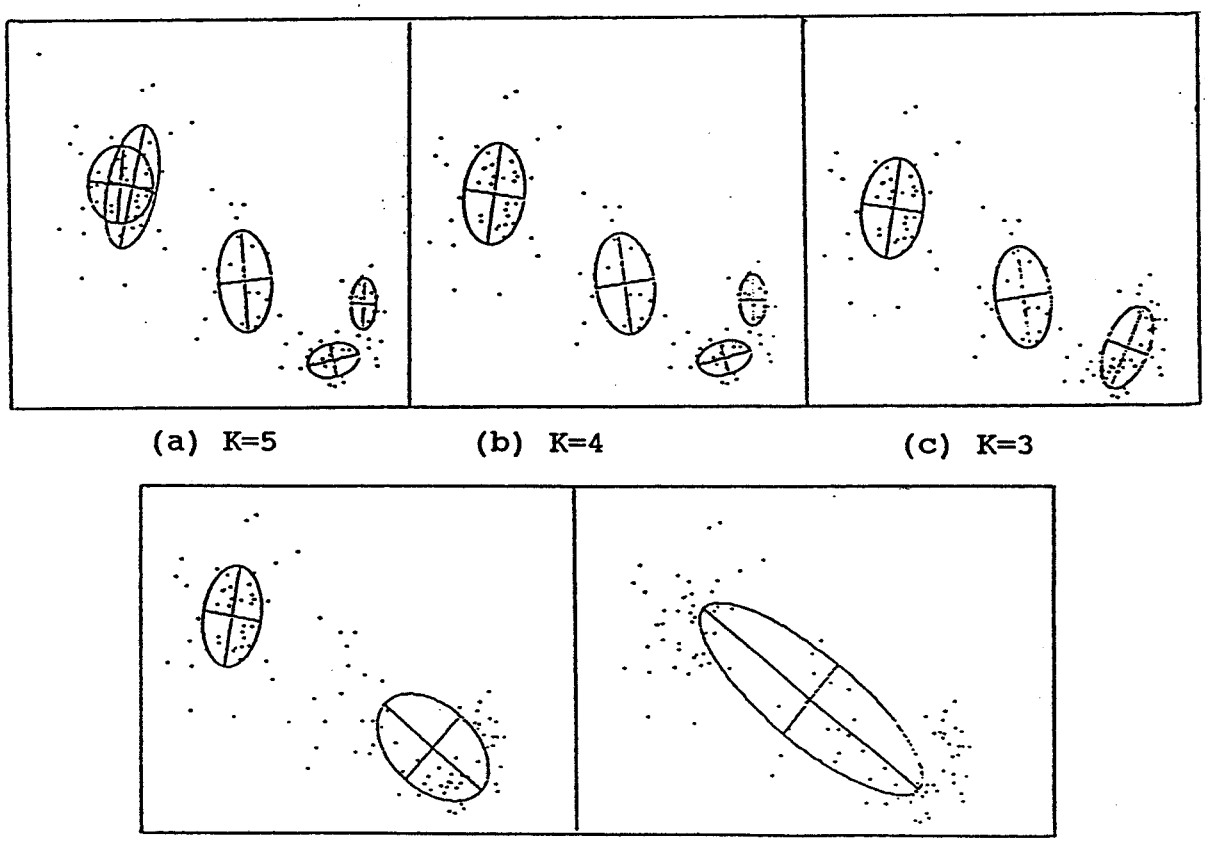


fig-1.26 déroulement de la fusion pour ex2

K	5	4	3	2	1
S(K)	2.02	0.478	0.425	0.375	0
C(K)	0.613	0.601	0.623	0.598	0.805
I(K)	14.28	12.49	12.89	12.34	16.10

tableau 1.10 résultats numériques de fusion pour ex2

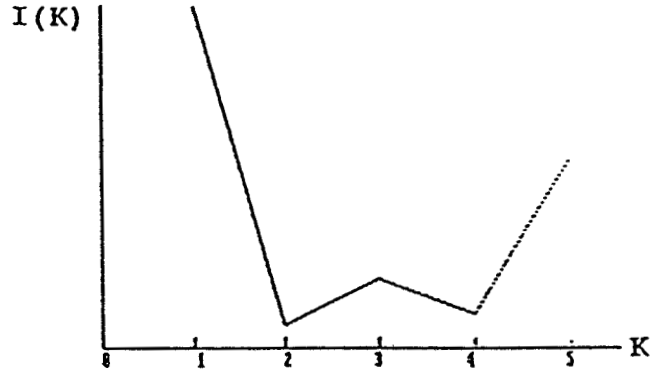
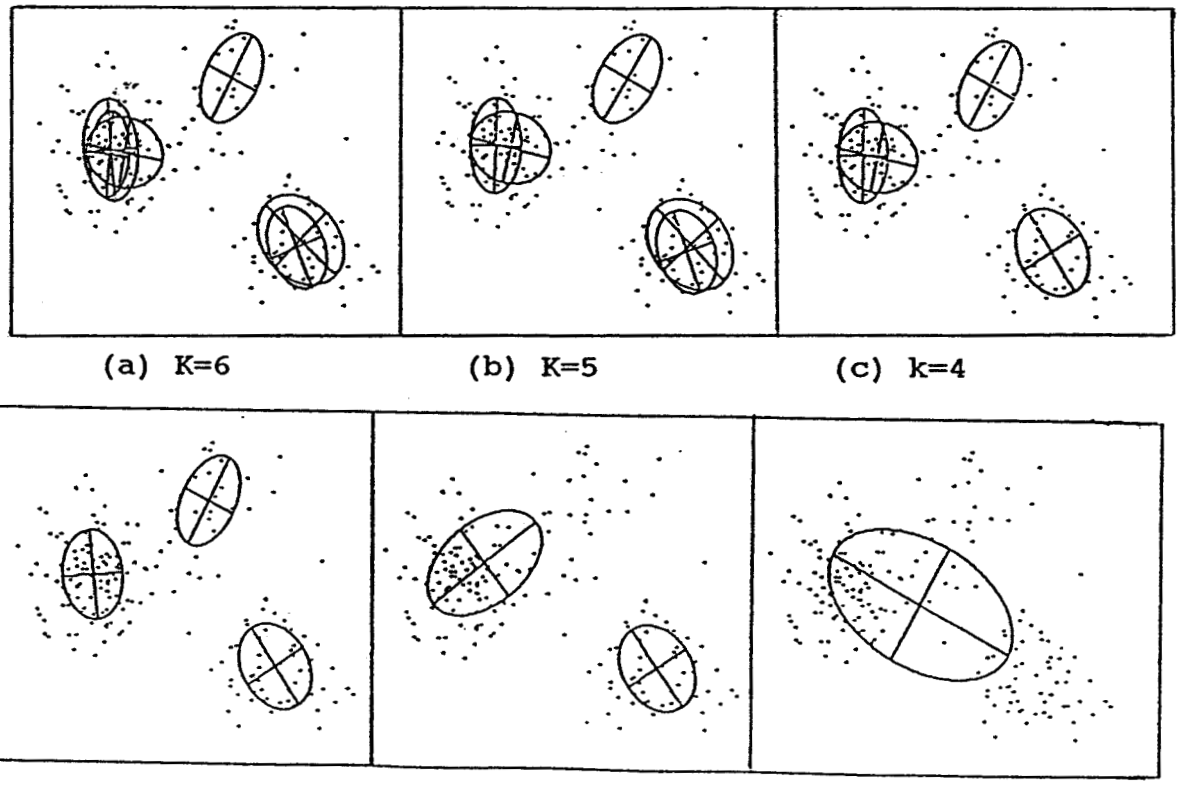


fig-1.27 évolution de l'indice au cours de la fusion (ex2)

Selon la fig-1.27, la fig-1.26(d) est la meilleure partition correspondant à K=2.

Exemple 3 (d=2, k_{max}=6, β=20):



(a) K=6 (b) K=5 (c) k=4

(d) K=3 (e) K=2 (f) K=1

fig-1.28 déroulement de la fusion pour ex3

K	6	5	4	3	2	1
S(K)	9.346	5.259	2.434	0.447	0.381	0
C(K)	0.629	0.634	0.625	0.614	0.606	0.687
I(K)	21.93	17.94	14.92	12.72	12.49	13.74

tableau 1.11 résultats numériques de fusion pour ex3

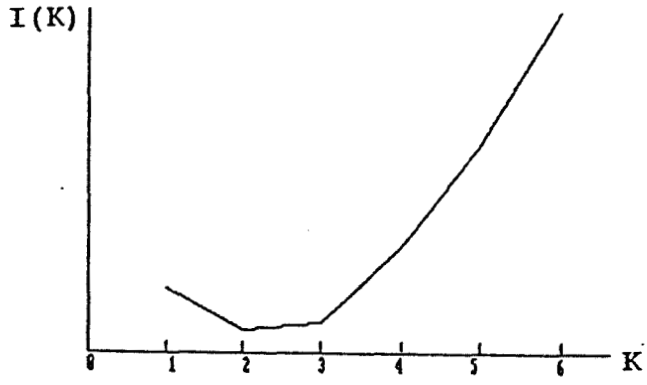


fig-1.29 évolution de l'indice au cours de la fusion (ex3)

La meilleure partition pour l'Exemple 3 est donnée fig-1.28(e) avec K=2.

Remarque: Dans l'Exemple 2, les valeurs des indices I(2) et I(4) sont très proches (cf fig-1.26). Dans ce cas, le choix de la meilleure partition est multiple (K=2 ou K=4).

IV.3.2 DISCUSSION SUR L'ALGORITHME PROPOSE

A partir de ces exemples, on peut constater que l'algorithme proposé peut effectivement fusionner des classes initialement différenciées. L'indice interne défini a priori peut réflèter une situation correcte de séparabilité et de compacité pour toutes les classes de chaque partition.

En général, cette méthode possède les avantages suivants par rapport aux autres méthodes de fusion:

- simplification du calcul effectué pour chaque fusion.

En effet, les autres méthodes conduisent souvent à des calculs très lourds sur chaque échantillon. Cette méthode, basée sur l'hypothèse d'une distribution normale, n'effectue pas directement d'opérations sur les échantillons individuels (sauf pour le calcul des compacités). Elle fusionne les classes selon les paramètres statistiques estimés à partir des échantillons

d'observation.

- obtention de la meilleure partition en tenant compte de tous les échantillons.

Dans certaines méthodes de fusion, les critères définis a priori conduisent souvent à des maxima ou minima locaux, correspondant à des meilleures partitions locales. On n'arrive donc pas à trouver la meilleure partition à partir d'un état initial quelconque. L'indice interne, défini précédemment, peut prendre en compte tous les échantillons caractérisés par les paramètres statistiques. La valeur minimale de cet indice correspond à la meilleure partition pour tous les états qui évoluent à partir d'un état initial quelconque.

Dans l'algorithme proposé, le paramètre β est sélectionné de façon manuelle par l'utilisateur. En effet, ce paramètre traduit le poids entre le degré de séparabilité et le degré de compacité. Il risque de conduire à une meilleure partition incorrecte s'il n'est pas adapté à la situation courante de distribution des données dans l'espace multi-dimensionnel. La Section IV.3.3 présente un algorithme de sélection de β de manière automatique à travers une procédure d'apprentissage.

IV.3.3 SELECTION DE β PAR APPRENTISSAGE

Pour déterminer une valeur raisonnable du paramètre β, un ensemble d'exemples, constituant une base d'apprentissage, est fourni a priori. Dans chacun des exemples, la meilleure partition est connue. Dans ces conditions, la valeur de β est progressivement apprise par exécution de la procédure de fusion pour tous les exemples de la base d'apprentissage.

Le principe de l'algorithme de sélection de β par apprentissage d'un exemple est suivant:

Soit m le nombre de classes de la meilleure partition pour l'exemple présenté et $]b_0, b_1[$ un intervalle initial de β.

On calcule ensuite respectivement la séparabilité et la compacité pour un ensemble de partitions (y compris la meilleure partition K=m) générées par la procédure de fusion. Deux listes de S(K) et C(K) sont obtenues. (K=1, 2, ..., m, ...).

Puisque l'état K=m représente la meilleure partition, on a: $I(K)>I(m)\ \forall K\neq m$. Ceci peut se traduire par les inégalités suivantes:

$$\begin{cases} \beta < \dfrac{S(K)-S(m)}{C(m)-C(K)} & \text{si } C(m)>C(K) \\ \beta > \dfrac{S(m)-S(K)}{C(K)-C(m)} & \text{si } C(m)<C(K) \end{cases} \qquad (1.4.22)$$

Dès lors, les valeurs de b_0 et b_1 peuvent se modifier de la façon suivante:

$$\begin{cases} b_0 := \max\{b_0, q_1, \ldots, q_r\} \\ b_1 := \min\{b_1, q_{r+1}, \ldots, q_{K-1}\} \end{cases} \qquad (1.4.23)$$

où $\{q_1, \ldots, q_r\} = \{\dfrac{S(m)-S(K)}{C(K)-C(m)} \mid C(m)<C(K)\}$

et $\{q_{r+1}, \ldots, q_{K-1}\} = \{\dfrac{S(K)-S(m)}{C(m)-C(K)} \mid C(m)>C(K)\}$

Si $b_0 > b_1$ ou si la meilleure partition défini a priori n'est pas rencontrée au cours de la procédure de fusion, alors K=m ne peut pas être considéré comme la meilleure partition. Dans ce cas, l'algorithme est défaillant et il faut redéfinir une meilleure partition de l'exemple.

Sinon, on obtient un nouvel intervalle de β: $]b_0, b_1[$.

Selon le principe précédent, l'algorithme de sélection de β par apprentissage d'un ensemble d'exemples se déroule de la manière suivante:

ETAPE 1 (Initialisation):

1.1 Prendre un ensemble d'exemples $E=\{e_1, e_2, \ldots, e_g\}$ comme base d'apprentissage.

1.2 Initialiser $]b_0, b_1[$ par $b_0 := 0$ et $b_1 := \infty$.

1.3 Déterminer le nombre maximal d'itérations de la procédure de fusion nb.

ETAPE 2: Modification de b_0 et b_1 par apprentissage de E.

Pour$_1$ i=1 jusqu'à g

Faire$_1$

2.1 Prendre l'exemple e_i de la base E.

2.2 Fixer la meilleure partition de e_i : K=m.

2.3 Modification de b_0 et b_1 par apprentissage de e_i .

$Pour_2$ j=1 jusqu'à nb

$Faire_2$

2.3.1 A partir de la partition K=m, chaque classe est divisée, de manière aléatoire, en 2 sous classes d'échantillons. On obtient alors une nouvelle partition initiale avec K=2m.

2.3.2 A partir de l'état K=2m, exécuter la procédure de fusion pour obtenir deux listes de S(K) et C(K) (K=2m, 2m-1, ..., 2, 1).

2.3.4 Modifier les valeurs de b_0 et b_1 selon l'équation (1.4.23).

2.3.5 Si $b_0 > b_1$ ou K=m n'est pas rencontré au cours de fusion, retourner à l'étape 2.2.

Fin_Faire_2

Fin_Pour_2

Fin_Faire_1
Fin_Pour_1

Etape 3: Après les itérations précédentes, on obtient l'intervalle final de $]b_0, b_1[$. Finalement, on choisit $\beta := (b_0+b_1)/2$.

Si on prend les 3 exemples précédents comme base d'apprentissage, on obtient les résultats ci-dessous:

Pour l'Exemple 1, si on choisit m=1, on obtient $\beta \in]0, 115[$.

Pour l'Exemple 2, si on choisit m=2, on obtient $\beta \in]1.8, \infty[$.

Pour l'Exemple 3, si on choisit m=2, on obtient $\beta \in]4.7, \infty[$.

Ainsi, après apprentissage de ces 3 exemples, la valeur de β se trouve dans l'intervalle $]4.7, 115[$. Dans la simulation précédente, on choisit β=20. Ceci correspond aux meilleures partitions m=1, m=2 et m=2 dans ces 3 exemples.

La procédure d'apprentissage basée sur l'algorithme de fusion permet de définir automatiquement le poids entre la séparabilité générale et la compacité générale selon les situations rencontrées. A partir de cette procédure, on obtient un indice de fusion correct. Cet indice peut être appliqué dans des situations similaires pour déterminer la meilleure partition des échantillons d'observation.

IV.4 ALGORITHME DE SCISSION

IV.4.1 PRINCIPE DE LA PROCEDURE D'ECLATEMENT D'UNE CLASSE

Comme pour la procédure de fusion, nous faisons les hypothèses suivantes:

- Chacune des classes à éclater contient un nombre suffisant d'échantillons pour faire évoluer une fenêtre à l'intérieur de cette classe. Le cas des échantillons isolés n'est pas envisagé dans cette approche.

- Chacune des classes éclatées respecte les lois de distribution normale.

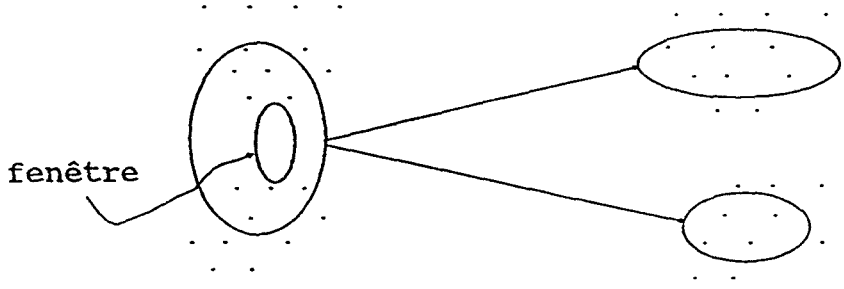


fig-1.30 éclatement d'une classe en deux sous classes

S'il existe 2 agglomérations d'échantillons dans une classe, nous souhaitons que chacune des classes éclatées représente une agglomération (cf fig-1.30).

IV.4.2 EVOLUTION DE FENETRE DANS UNE CLASSE

Pour mettre en évidence l'éclatement, on fait évoluer une fenêtre parmi les échantillons de la classe à éclater. En observant les effets de l'évolution de cette fenêtre, on peut tester si elle s'approche d'une agglomération d'échantillons.

Cette procédure ressemble au glissement de fenêtre entre 2 classes présenté dans la Section II.4.3. En appliquant les conclusions du Chapitre II, on propose la méthode d'éclatement d'une classe C_k selon la procédure suivante:

Supposons que la taille de la fenêtre est $n<N_k$ (N_k est le nombre d'échantillons dans C_k et C_k comporte 2 agglomérations d'échantillons, notées A_1 et A_2). Nous rappelons ci-dessous les

éléments de la Section II.4.3:

Le nombre d'échantillons de A_2 appartenant à la fenêtre est $i \leqslant n$.

Le nombre d'échantillons de A_1 appartenant à la fenêtre est $n-i$.

Alors l'estimation du vecteur moyenne de la fenêtre est:

$$M(i) = \frac{1}{n}\sum_{j=1}^{n} Y_j$$

où $Y_1,\ldots,Y_n$ sont les échantillons appartenant à la fenêtre.

L'espérance mathématique de cette estimation s'écrit de la façon suivante (cf l'équation (1.2.33)):

$$E(M(i)) = M_1 + \frac{1}{n}(M_2 - M_1)\,i$$

où M_1, M_2 sont respectivement les vecteurs moyennes de A_1 et A_2.

Après une opération de retrait d'un échantillon de la fenêtre et d'ajout d'un échantillon extérieur mais appartenant également à la classe que l'on souhaite éclater, on obtient les 3 cas possibles ci-dessous:

1) Si l'échantillon retiré et l'échantillon ajouté appartiennent à la même agglomération (A_1 ou A_2) de la classe C_k, le vecteur moyenne ne change pas après cette opération. La variation du vecteur moyenne s'écrit alors $E(\Delta M(i))=0$.

2) Si l'échantillon retiré appartient à A_1 et l'échantillon ajouté appartient à A_2, le vecteur moyenne devient $E(M(i+1))$ et la variation du vecteur moyenne est la suivante:

$$E(\Delta M(i)) = E(M(i+1)) - E(M(i)) = \frac{1}{n}(M_2 - M_1) \qquad (1.4.24)$$

3) Si l'échantillon retiré appartient à A_2 et l'échantillon ajouté appartient à A_1, le vecteur moyenne devient $E(M(i-1))$ après l'opération. La variation du vecteur moyenne est la suivante:

$$E(\Delta M(i)) = E(M(i-1)) - E(M(i)) = -\frac{1}{n}(M_2 - M_1) \qquad (1.4.25)$$

Selon l'analyse précédente, on peut tester l'appartenance de chaque échantillon de la fenêtre à A_1 ou A_2 et éclater cette fenêtre en 2 sous ensembles. Pour cela on procède de la façon suivante:

Dans la fenêtre $\{Y_1, \ldots, Y_n\}$, on retire Y_1 et on ajoute un échantillon Y_{n+1}. On obtient une estimation du nouveau vecteur moyenne $M'(i)$. La variation $E(\Delta M(i))$ peut être estimée par $\Delta M(i)=M'(i)-M(i)$.

Si $\|\Delta M(i)\|<\varepsilon$ (ε est un réel positif petit), on considère que $\Delta M(i)=0$ et que Y_1 et Y_{n+1} appartiennent à la même agglomération.

Si $\|\Delta M(i)\|>\varepsilon$, on considère que Y_1, Y_{n+1} appartiennent à deux agglomérations différentes. Dans ce cas, l'estimation de $E(\Delta M(i))$ est noté par $\Delta E=\Delta M(i)$.

Si $Y_1 \in A_1$, ΔE peut être considéré comme une estimation de $\frac{1}{n}(M_2-M_1)$;
Si $Y_1 \in A_2$, ΔE peut être considéré comme une estimation de $-\frac{1}{n}(M_2-M_1)$.

Après l'obtention de ΔE, Y_{n+1} est retiré et Y_1 est récupéré par la fenêtre.

On répète l'opération précédente pour tout autre échantillon de la fenêtre: Y_j ($j \in \{2,\ldots,n\}$). Pour cela, on retire Y_j et on ajoute un échantillon Y'_{n+1} extérieur à la fenêtre et tel que $\|\Delta M(i)\|>\varepsilon$. Evidemment, $\Delta M(i)$ peut également être considéré comme une estimation de $\frac{1}{n}(M_2-M_1)$ si $Y_j \in A_1$ et $-\frac{1}{n}(M_2-M_1)$ si $Y_j \in A_2$.

Comme précédemment, Y_j est récupéré par la fenêtre et Y'_{n+1} est retiré après l'opération.

Ainsi, on peut énoncer les conditions de discrimination suivantes illustrées fig-1.31:

si $\Delta M(i)\cdot\Delta E>0$ Y_j et Y_1 appartiennent au même sous ensemble de C_k (A_1 ou A_2) (cf fig-1.31(b))

si $\Delta M(i)\cdot\Delta E<0$ Y_j et Y_1 appartiennent à deux sous ensembles différents de C_k ($Y_j \in A_1$ et $Y_1 \in A_2$ ou $Y_j \in A_2$ et $Y_1 \in A_1$) (cf fig-1.31(a)).

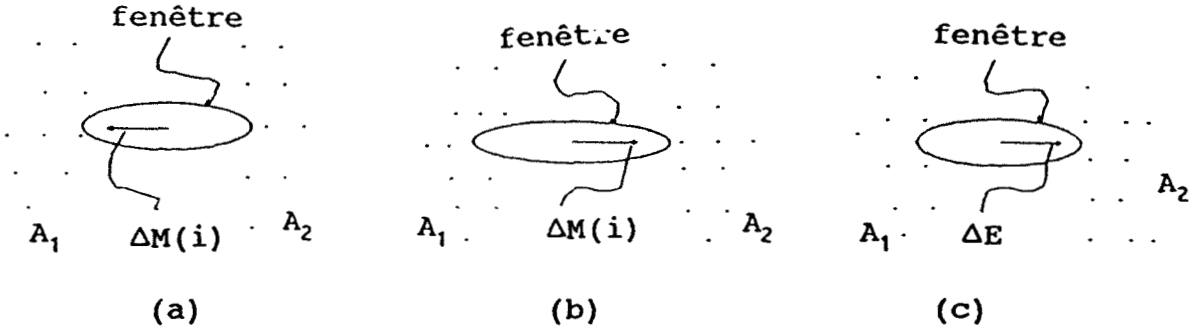


fig-1.31 discrimination des appartenances des échantillons de la fenêtre

De cette manière, on teste si chaque échantillon de la fenêtre appartient à la même classe que Y_1 et on peut diviser cette fenêtre en deux sous ensembles E_1 et E_2.

Pour obtenir un effet visible de l'évolution de la fenêtre, la différence entre M_1 et M_2 doit être suffisamment grande et le choix de n doit être raisonnable.

Si n est trop grand, la variation $\Delta M(i)$ est trop petite et les 3 cas possibles présentés ci-dessus ne peuvent pas être différentiés à partir de $\Delta M(i)$.

Si n est trop petit, la précision d'estimation pour le vecteur moyenne est insuffisante et les échantillons de A_1 et A_2 ne peuvent pas être effectivement classés à partir de la fenêtre.

IV.4.3 ALLOCATION DES ECHANTILLONS DANS E1 ET E2

A partir des deux sous ensembles E_1 et E_2, on teste l'appartenance de tous les échantillons de C_k et on intégre ces échantillons dans E_1 ou dans E_2. La procédure correspondant à cette opération est présenté ci-dessous:

Soient Z_1 et Z_2 les vecteurs moyennes de E_1 et E_2. On prend un échantillon de C_k à l'extérieur de la fenêtre, noté Y_{n+1}. Ensuite, on calcule respectivement les distances euclidiennes entre Y_{n+1} et Z_1 et entre Y_{n+1} et Z_2, notés d_1 et d_2.

Si $d_1 < d_2$, alors $Y_{n+1} \in E_1$

Si $d_2 > d_1$, alors $Y_{n+1} \in E_2$

Après avoir décidé de l'appartenance de Y_{n+1}, on ajoute Y_{n+1} à E_1 ou E_2 et on calcule, selon la récurrence positive de

l'algorithme rapide, le nouveau vecteur moyenne et la nouvelle matrice de covariance. Cette procédure se répète jusqu'à ce que tous les échantillons de C_k soient classés.

IV.4.4 RECHERCHE DE LA MEILLEURE PARTITION PAR SCISSION-FUSION

A partir d'un mélange de nuages d'échantillons, on peut obtenir une partition de multi-classes en répétant l'algorithme d'éclatement d'une classe en deux. Après arrangement des classes possédant un petit nombre d'échantillons (l'objectif de cette opération est de supprimer les classes mal éclatées qui

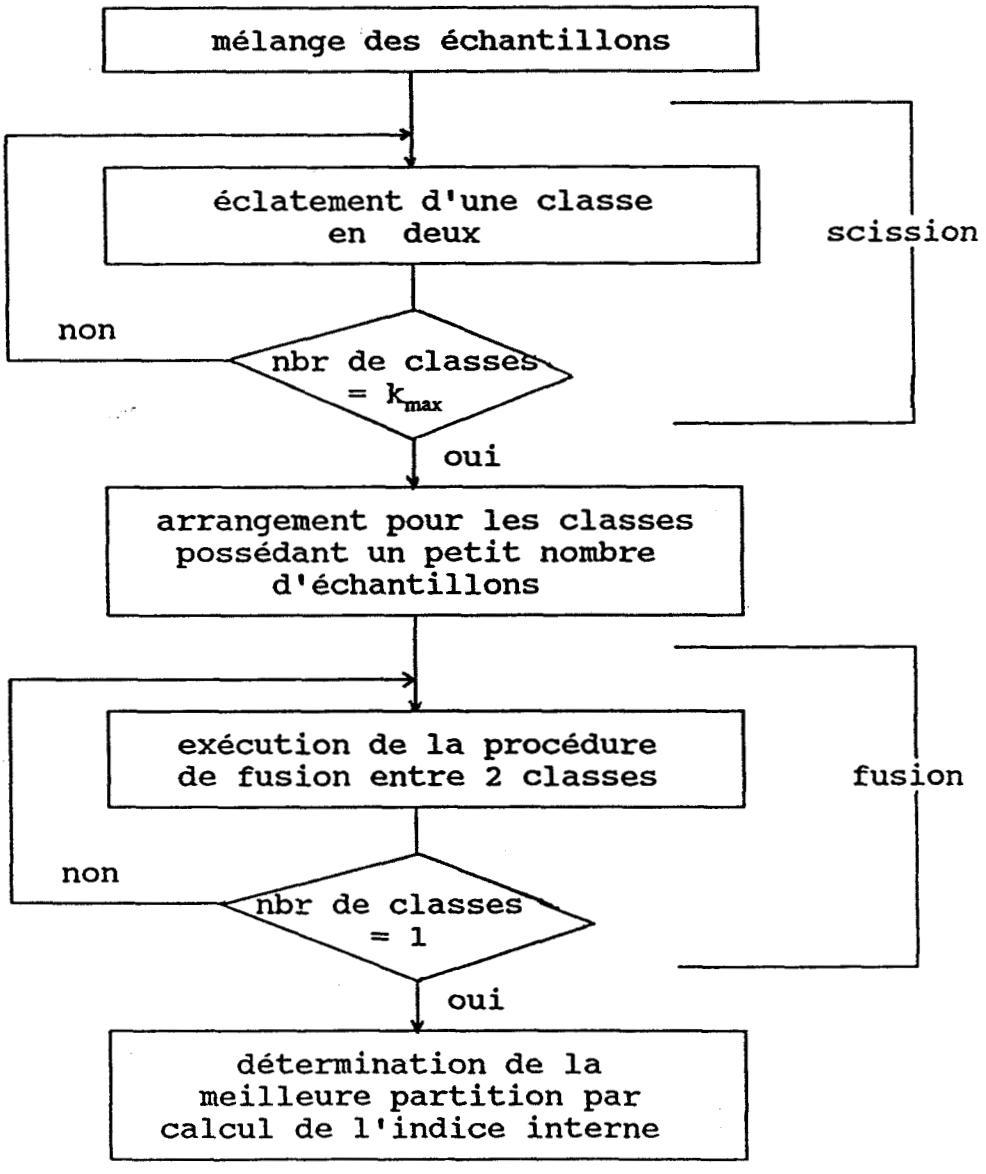


fig-1.32 procédure de recherche de la meilleure partition par scission-fusion

contiennent des échantillons appartenant à des agglomérations différentes, cf IV.4.6), on exécute la procédure de fusion présentée dans IV.2 pour rechercher la meilleure partition par calcul de l'indice interne (cf fig-1.32).

Dans la fig-1.32, k_{max} est le nombre maximal des classes générées par la procédure de scission. Il est défini a priori.

Dans l'algorithme de la fig-1.32, il reste 2 problèmes non résolus par les procédures présentées ci-dessus:

- Comment choisir la classe à éclater parmi plusieurs classes d'échantillons?

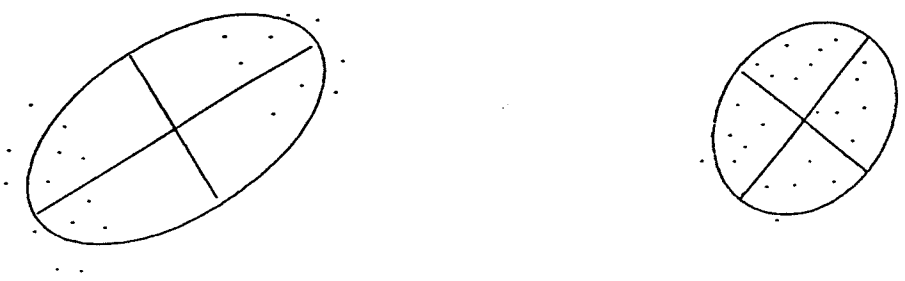
- Comment arranger les classes qui possèdent un petit nombre d'échantillons?

IV.4.5 CHOIX DE LA CLASSE A ECLATER

Dans la procédure de scission illustrée fig-1.32, s'il existe au moins deux classes d'échantillons, il est nécessaire de décider quelle classe doit être éclatée en deux sous classes.

L'objectif de la scission est de regrouper raisonnablement les échantillons dans des classes différentes. On souhaite donc que la procédure d'éclatement puisse séparer une classe en deux agglomérations différentes.

Selon l'analyse précédente, on doit choisir une classe où la possibilité d'avoir plusieurs agglomérations d'échantillons est la plus grande. Dans cette situation, le grand axe de l'hyper-ellipse de cette classe est souvent plus grand que ceux des autres classes.



(a) 2 agglomérations (b) 1 agglomération

fig-1.33 choix de la classe à éclater

Dans la fig-1.33, le grand axe de (a) est plus long que celui de (b). On choisit donc (a) comme la classe à éclater.

Pour une classe C_k, le grand axe se calcule de la façon suivante:

$$AM(C_k) = \max_{1\leqslant i\leqslant d} \{\sqrt{\lambda_{k\,i}}\} \qquad (1.4.26)$$

où les $\{\lambda_{k\,i}\}$ (i=1,2,...,d) sont valeurs propres de C_k.

IV.4.6 ARRANGEMENT POUR LES CLASSES COMPORTANT UN PETIT NOMBRE D'ECHANTILLONS

Pour le 2ème problème, on commence par analyser les classes possédant un petit nombre d'échantillons.

En effet, la procédure d'éclatement est adaptée au cas où le nombre d'échantillons dans une classe est suffisamment grand. Dans le cas contraire, l'algorithme d'éclatement ne peut pas fonctionner car le fenêtrage n'est pas possible.

D'après l'hypothèse dans IV.4.1, chaque agglomération contient un nombre suffisant d'échantillons. Ainsi, la procédure d'arrangement doit supprimer les classes possédant un petit nombre d'échantillons et réassigner leurs échantillons aux classes voisines.

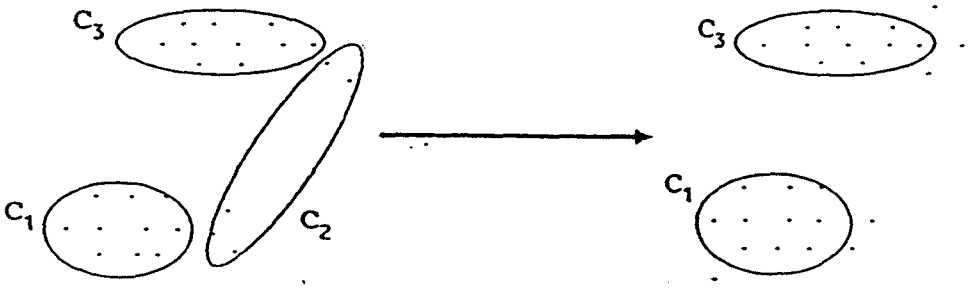


fig-1.34 arrangement de la classe C_2

Par exemple, dans la fig-1.34, le nombre d'échantillons de C_2 est trop petite (N_2<n). Il est nécesaire de réassigner les échantillons de C_2 aux classes voisines. Après arrangement, ses échantillons sont assignés respectivement à C_1 et C_3.

En général, si C_k contient des échantillons appartenant à différentes agglomérations, son grand axe est plus grand que ceux des autres classes. En ordonnant les grand axes pour toutes les classes, on propose la procédure d'arrangement de la manière suivante:

Etape 1: Supposons qu'il existe k_{max} classes d'échantillons après exécution de la procédure de scission, ces classes sont divisées en deux sous ensembles F_1 et F_2:

$F_1 = \{C_1, \ldots, C_l\}$
$= \{C_k \mid$ le nombre d'échantillons dans $C_k < n\}$

$F_2 = \{C_{l+1}, \ldots, C_{k_{max}}\}$
$= \{C_k \mid$ le nombre d'échantillons dans $C_k \geqslant n\}$ où $1 \leqslant l \leqslant k_{max}$

Si F_1 est vide, l'algotithme s'arrête; sinon, aller à l'étape 2.

Etape 2: Calculer les grands axes des classes de F_1: $AM(C_1)$, $AM(C_2)$, ..., $AM(C_l)$.

En notant $AM(F_1)=\max\{AM(C_1), AM(C_2), \ldots, AM(C_l)\}$

rechercher dans F_1 une classe C_k qui satisfait les conditions suivantes:

$C_k \in F_1$, $AM(C_k)=AM(F_1)$

Etape 3: Pour chaque échantillon Y de C_k, calculer ses distances de MAHALOBBI aux vecteurs moyennes des autres classes:

$d_i = (Y-M_i)^T S_i^{-1} (Y-M_i)$ $\quad i \in \{1, \ldots, k_{max}\}$ et $i \neq k$

Choisir la plus petite valeur d_j parmi les d_i précédents et assigner Y à la classe C_j. Calculer les nouvelles caractéristiques de C_j (M_j et S_j) par la récurrrence positive de l'algorithme rapide.

Etape 4: Supprimer la classe C_k. Pour i=k jusqu'à $k_{max}-1$ Faire $C_i := C_{i+1}$. Assigner $k_{max} := k_{max}-1$ et aller à l'étape 1.

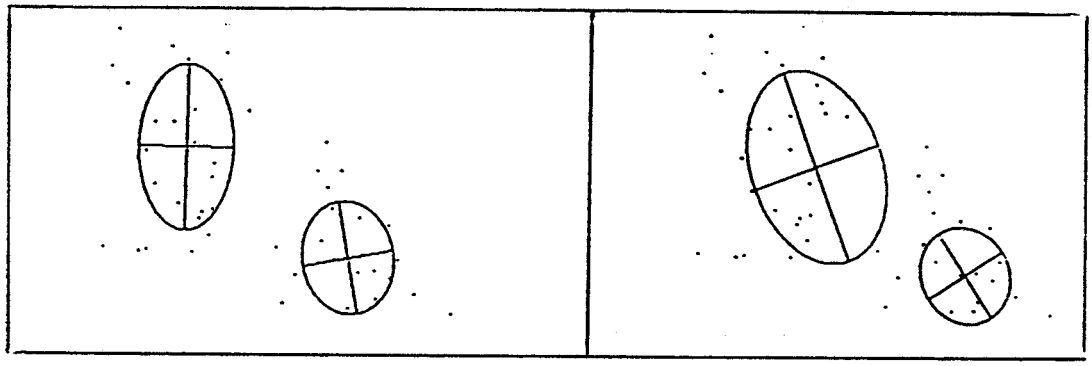
En utilisant les procédures présentées dans ce chapitre, on peut effectuer la recherche de la meilleure partition d'après le schéma de la fig-1.32.

IV.5 SIMULATION DE L'ALGORITHME DE SCISSION

IV.5.1 SIMULATION DE L'ECLATEMENT D'UNE CLASSE

On donne ci-dessous quelques exemples de simulation de la procédure d'éclatement d'une classe en 2. Dans chaque exemple, on commence par générer deux classes d'échantillons dans l'espace multi-dimensionnel. Ensuite, on mélange ces échantillons et on appelle la procédure d'éclatement présentée dans IV.4.1. Finalement, on effectue une comparaison entre les classes originales et les classes éclatées.

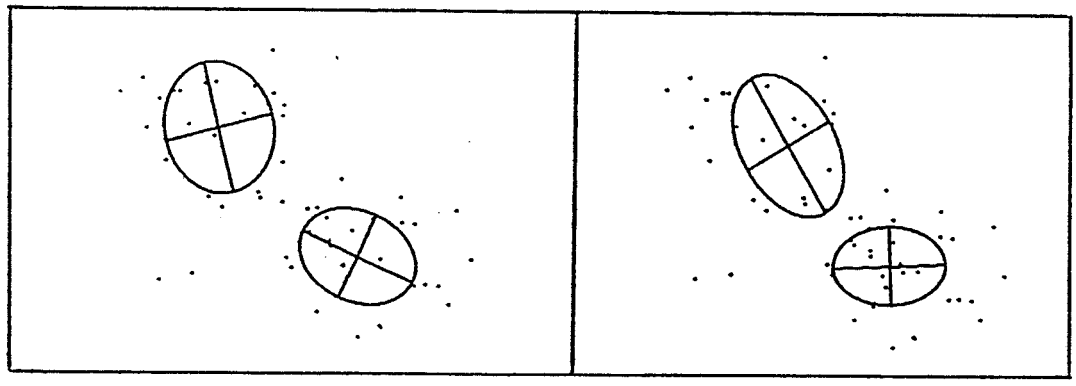
Exemple 4:



(a) classes originales (b) classes éclatées

fig-1.35 Simulation de la procédure d'éclatement (d=2)

Exemple 5:



(a) classes originales (b) classes éclatées

fig-1.36 Simulation de la procédure d'éclatement (d=2)

D'après les deux exemples ci-dessus, les formes des classes éclatées ressemblent à celles des classes originales.

Pour évaluer les performances de la procédure d'éclatement d'une classe, on définit les taux de classement correct des échantillons t_1 et t_2 de la façon suivante:

Supposons que C_1', C_2' sont les deux classes originales et C_1, C_2 sont les deux classes éclatées. On obtient:

$$t_1 = \max\left\{\frac{Card\{Y|Y\in C_1,\ Y\in C_1'\}}{Card\{Y|Y\in C_1\}},\ \frac{Card\{Y|Y\in C_1,\ Y\in C_2'\}}{Card\{Y|Y\in C_1\}}\right\} \qquad (1.4.26)$$

$$t_2 = \max\left\{\frac{Card\{Y|Y\in C_2,\ Y\in C_1'\}}{Card\{Y|Y\in C_2\}},\ \frac{Card\{Y|Y\in C_2,\ Y\in C_2'\}}{Card\{Y|Y\in C_2\}}\right\} \qquad (1.4.27)$$

Le tableau ci-dessous donne les deux paramètres t_1 et t_2 pour 3 exemples de simulation.

taux de classement	Exemple 4 (d=2)	Exemple 5 (d=2)	Exemple 6 (d=3)
t_1	0.90	0.97	0.93
t_2	1	0.89	0.95

tableau 1.12 taux de classement t_1 et t_2

Dans l'Exemple 4, 90% des échantillons de C_1 proviennent de la classe originale C_1' et 100% des échantillons de C_2 proviennent de la classe originale C_2'.

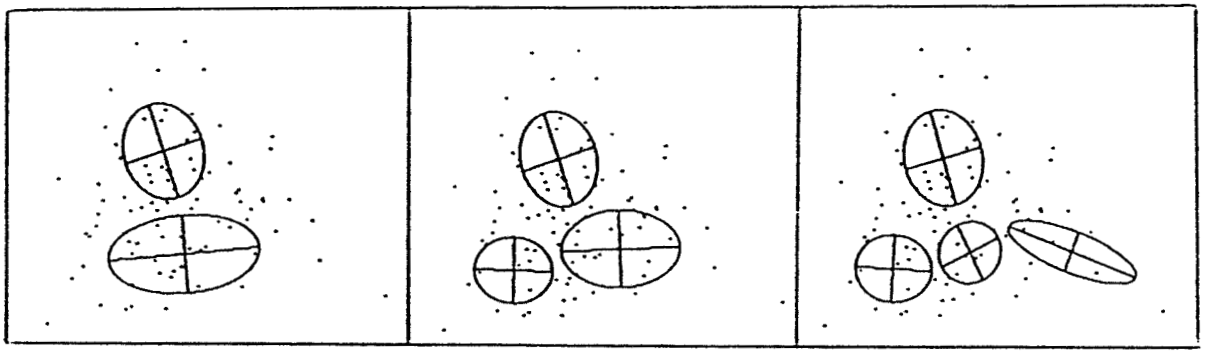
Dans les exemples 5 et 6, ces taux de classement sont également élevés. D'après l'analyse précédente, il apparaît que la procédure d'éclatement permet une assignation correcte d'échantillons provenant d'origines différentes.

IV.5.2 SIMULATION DE LA RECHERCHE DE LA MEILLEURE PARTITION PAR SCISSION-FUSION

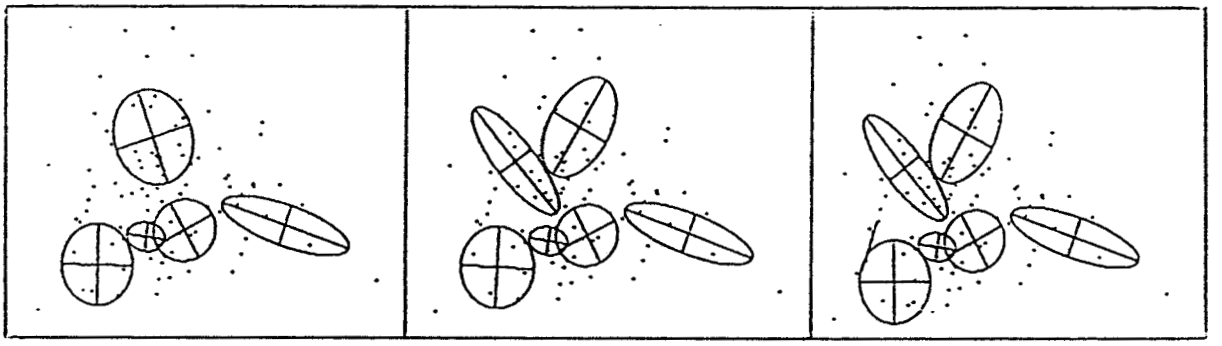
Pour chacun des Exemples 1, Exemple 2 et Exemple 3, on mélange les échantillons de toutes les classes originales et on exécute la procédure de scission jusqu'à ce que le nombre total de classes éclatées soit égal à k_{max}. Après arrangement des classes possédant un petit nombre d'échantillons, on appelle la procédure de fusion pour trouver une partition optimale correspondant à l'indice minimal (cf fig-1.32). Finalement, on effectue la comparaison entre la meilleure partition obtenue par

la méthode de fusion (cf IV.2) et la meilleure partition obtenue par la méthode de scission-fusion (cf IV.4). Le déroulement de l'algorithme de scission-fusion est illustré sur les figures ci-dessous:

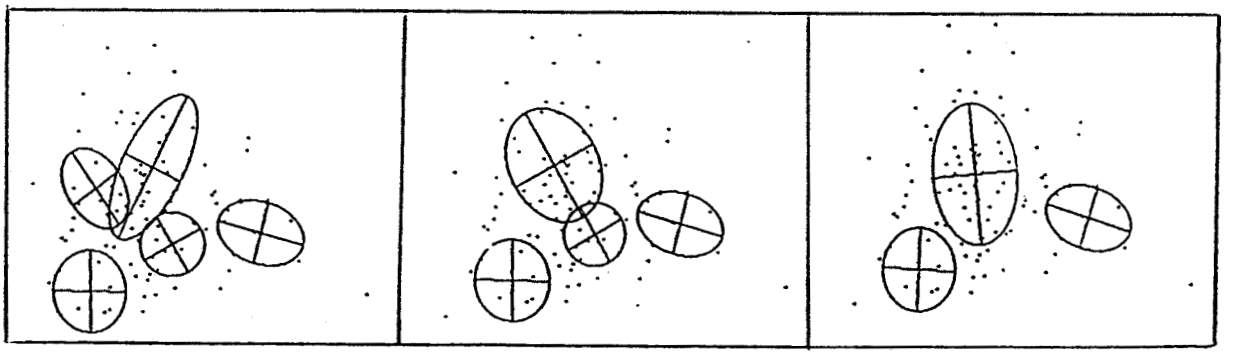
Pour l'Exemple 1 ($k_{max}=7$, $\beta=20$):



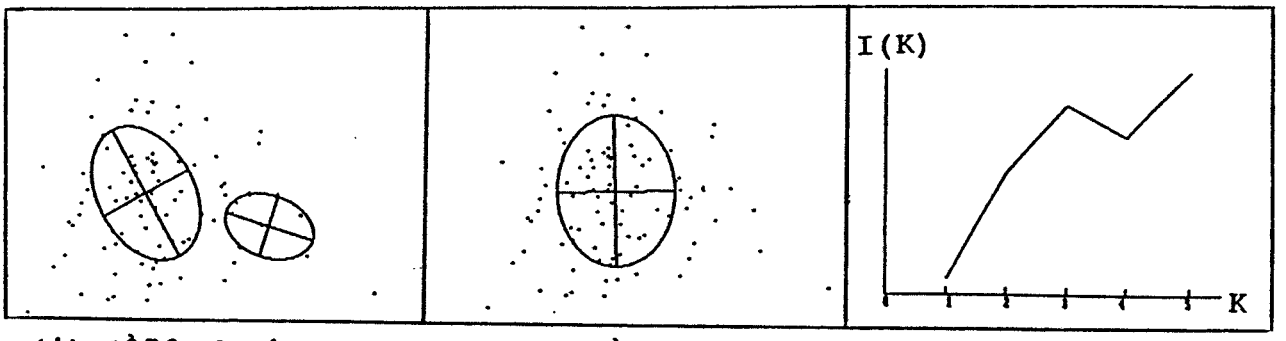
(a) 1er éclatement (K=2) (b) 2ème éclatement (K=3) (c) 3ème éclatement (K=4)



(d) 4ème éclatement (K=5) (e) 5ème éclatement (K=6) (f) 6ème éclatement (K=7)



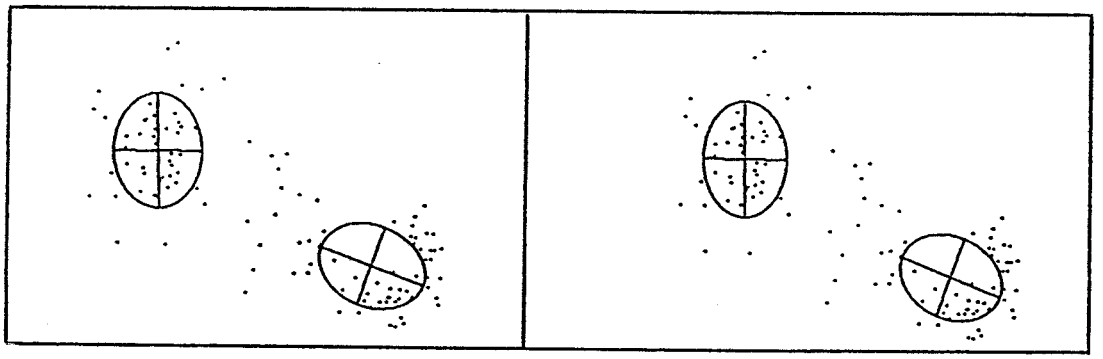
(g) arrangement des classes après la scission (K=5) (h) 1ère fusion (K=4) (i) 2ème fusion (K=3)



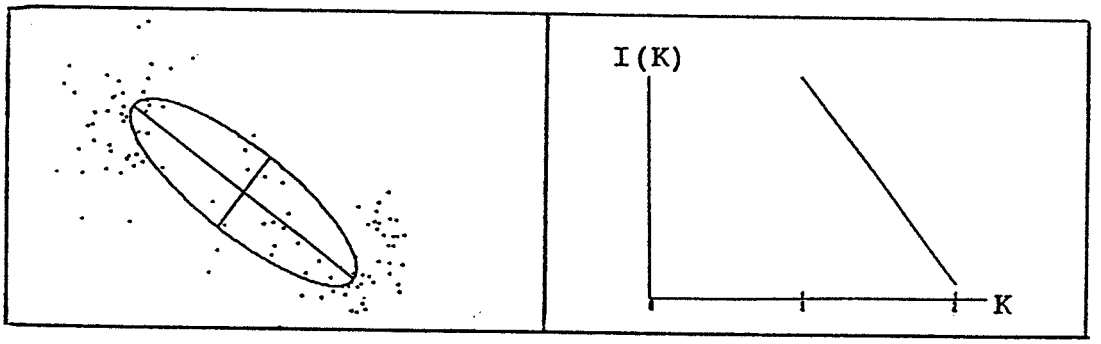
(j) 3ème fusion (K=2) (k) 4ème fusion (K=1) (l) indice interne

fig-1.37 recherche de la meilleure partition par scission-fusion (ex1)

Pour l'Exemple 2 (k_{max}=2, β=20):



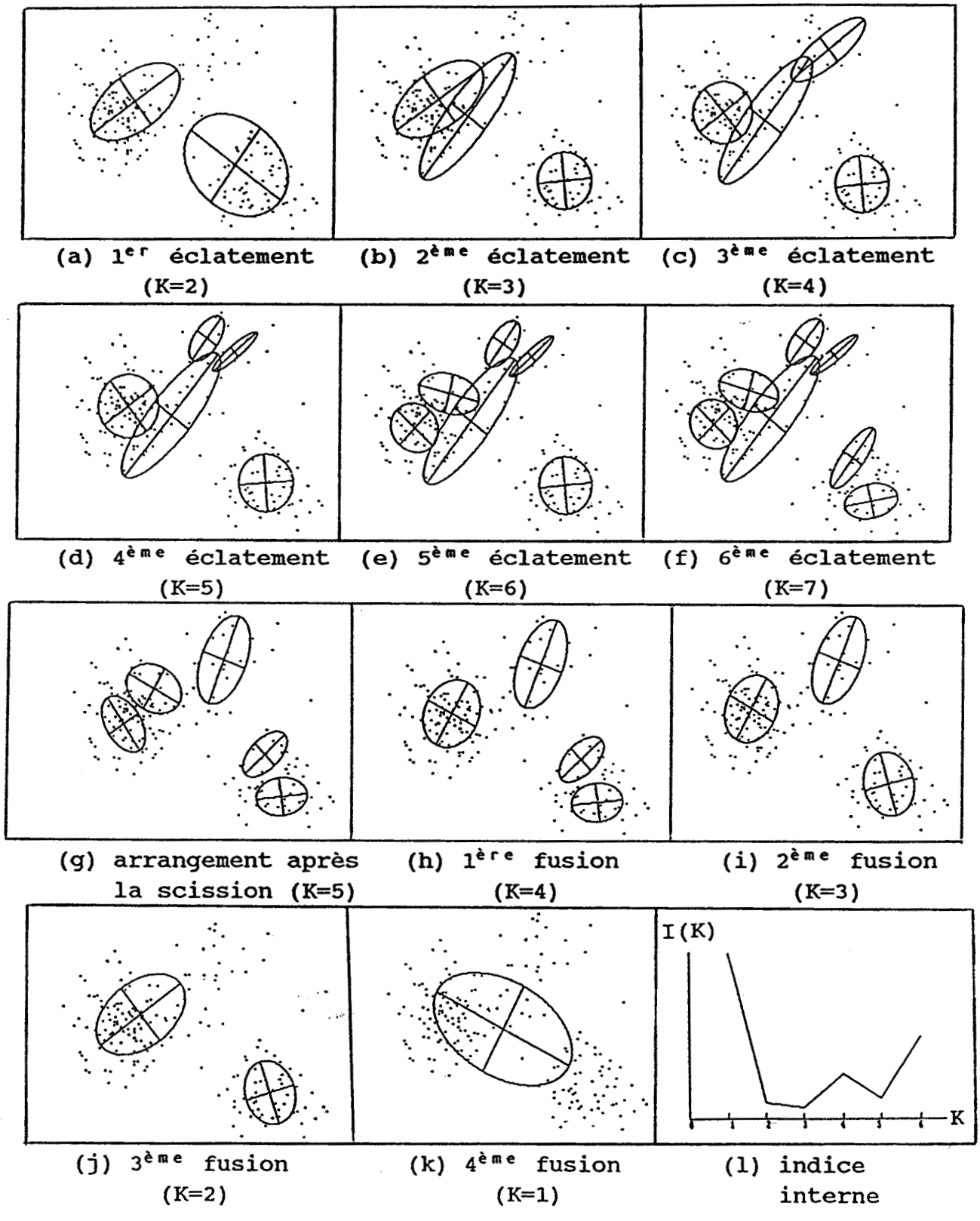
(a) 1er éclatement (K=2) (b) arrangement après la scission (K=2)



(c) 1ère fusion (K=1) (d) indice interne

fig-1.38 recherche de la meilleure partition par scission-fusion (ex2)

Pour l'Exemple 3 (k_{max}=7, β=20):



(a) 1er éclatement (K=2) (b) 2ème éclatement (K=3) (c) 3ème éclatement (K=4)

(d) 4ème éclatement (K=5) (e) 5ème éclatement (K=6) (f) 6ème éclatement (K=7)

(g) arrangement après la scission (K=5) (h) 1ère fusion (K=4) (i) 2ème fusion (K=3)

(j) 3ème fusion (K=2) (k) 4ème fusion (K=1) (l) indice interne

fig-1.38 recherche de la meilleure partition par scission-fusion (ex3)

La meilleure partition de l'exemple 1 correspond à la fig-1.36(k) avec K=1, ce qui est conforme au résultat de la méthode de fusion (IV.2). Dans les exemple 2 et 3, les

meilleures partitions correspondent respectivement à la fig-1.37(b) avec K=2 et à la fig-1.38(i) avec K=3. Ces résultats sont similaires à ceux de la procédure de fusion (IV.3). En général, la meilleure partition obtenue par l'algorithme de scission-fusion donne donc des résultats très précis.

IV.5.3 REMARQUES FINALES

L'algorithme présenté ci-dessus est basé sur l'évolution d'une fenêtre au sein de chaque classe à éclater. Si deux agglomérations dans une classe sont assez proches, l'effet de l'évolution de la fenêtre n'est pas évident et les échantillons de cette classe ne peuvent pas être correctement classés.

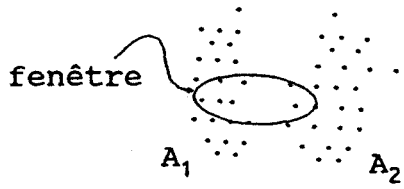


fig-1.39 classe difficile à éclater

Dans la procédure d'éclatement, on utilise la distance euclidienne pour classer les échantillons dans les différentes classes. En effet, dans ce cas, la distance de MAHALOBBI n'est pas disponible car les éléments propres (valeurs propres et vecteurs propres) de chaque classe sont inconnus.

D'après les exemples de simulation, les performances de l'algorithme dépendent des formes des nuages d'échantillons dans l'espace multi-dimensionnel. Le choix des paramètres (β, n, e, etc) est également important pour obtenir de bonnes performances de l'algorithme.

Dans la procédure de recherche de la meilleure partition, l'indice interne que nous avons défini joue un role essentiel. En effet, au cours de la procédure de fusion deux à deux, chaque état est caractérisé par cet indice.

Dans ce chapitre, nous faisons l'hypothèse que les échantillons respectent les lois de distribution normale. Sous cette hypothèse, la forme d'une classe peut être exprimée par une hyper-ellipse dans l'espace multi-dimensionnel. En introduisant l'équation de contour de l'hyper-ellipse dans la définition, la forme de la classe est prise en compte par l'indice interne.

CONCLUSION

Dans la première partie, on a établi le modèle adopté pour le processus dynamique étudié. Selon ce modèle, on analyse la structure statistique interne du processus à partir de l'observation des manifestations externes d'événements caractéristiques (estimation des paramètres statistiques, classification des échantillons, recherche de la meilleure partition des échantillons, etc).

Dans la pratique, on souhaite que l'observateur du processus puisse non seulement percevoir et interpréter les messages émis par le processus, mais aussi agir directement sur le processus pour le conduire vers une structure interne générant une sortie désirée. Dans ce cas, les manifestations externes du processus sont commandées par l'observateur.

Dans la deuxième partie du mémoire, on présente des principes de commande du processus sans établir de modèle mathématique. Deux méthodes de commande sont proposées.

BIBLIOGRAPHIE

[VAS 82] C.VASSEUR
"La notion d'événement dans les systèmes dynamiques: détection, classification temps réel et application à la conception d'une instrumentation distribuée"
Thèse de Doctorat d'Etat - Sciences Physiques, Lille I, juin 1982

[WAL 77] B.WALLISER
Systèmes et modèles, introduction critique à l'analyse des systèmes
Edition Seuil, 1977

[ASH 58] W.ASHBY
Introduction à la cybernétique
Paris, Dunod, 1958

[BER 73] L.V.BERTALANFFY
Théorie générale des systèmes
Paris, Dunod, 1973

[FER 67] T.S.FERGUSON
Mathematical statistics
1967 Academic Press N.Y. San Francisco London

[FOU 72] D.C.FOURGEAU et A.FUCHS
Statistique
Année 1972 - Academic Press

[DUD 73] R.U.DUDA et D.E.HART
Pattern classification and scene analysis
New York: Willey, 1973

[POS 87] J.G.POSTAIRE
De l'image à la décision
Paris, Dunod, 1987

[FUF 78] K.FUFUNAGA et R.D.SHORT
"Nonlinear feature extraction with a general criterion function"
IEEE Trans. Inform. Theory vol.IT-24, no.5, 1978

[FUK 80] K.FUKUNAGA et R.D.SHORT
"A class of feature extraction criteria and its relation to the bayes risk estimate"
IEEE Trans. Inform. Theory, vol.IT-26, no.1, 1980

[FUK 72] K.FUKUNAGA
Introduction to statistical pattern recognition
New York: Academic, 1972

[EBE 73] A.EBERHARD
"An optimal discrete window for the calculation of power spectra"
IEEE Trans. Audio Electroacoust. Au21, 37-43, 1973

[DOD 77] G.DODENSTEIN et H.M.PRAETORIUS
"Feature extraction from the electroencephalogram by adaptive segmentation"
Proceedings of the IEEE, 65. 642-652, May 1977

[CHE 83] B.CHEBEL
"Application de l'analyse de structure et de la fonction entropie au traitement du signal"
Thèse de Doctorat-Ingénieur, Lille, mars 1983

[AND 58] T.W.ANDERSON
Introduction to multivariate statistic
John Wimey, N.T., 1958

[KAZ 77] D.KAZAKOS
"Récursive estimation of a prior probabilities using a mixture"
IEEE Trans. Inform. Theory vol.23, mars 1977

[YON 70] T.Y.YONG et C.CORALUPPI
"Stochastic estimation of a mixture of normal density function on using an information theory criterion"
IEEE Trans. Inform. Theory vol.16, May 1970

[FUK 83] K.S.FUKUNAGA et T.E.FLICK
"Estimation of the parameters of a gaussian mixture using the method of moments"
IEEE Trans. Pami. vol.5, n°4, July 1983

[POS 81] J.G.POSTAIRE et C.VASSEUR
"An approximation solution to normal mixture identification with application to un superised pattern classification"
IEEE Trans. Pami. vol.3 mars 1981

[LAK 87] R.LAKEL
"Contribution à l'identification des mélanges gaussiens à l'aide d'une approche récursive"
Thèse de Docteur Ingénieur, Lille I, janvier 1987

[GU 85] T.GU et B.DUBUISSON
"A general model of pattern classification"
RAIRO pp.69-91, 1985

[VAS 90] C.VASSEUR, R.LAKEL et P.HUBLIN
"Estimation récursive des caractéristiques d'une classe en reconnaissance des formes"
RAIRO, vol.24, n°6, 1990

[TIK 67] M.L.TIKU
"Estimating the mean and standard deviation from a censored normal sample"
Biometrica, 54, 1 and 2, p155, 1967.

[AST 80] K.J.ASTRÖM
"Maximum likelihood and prediction error methods"
Automatica, 16, 551-574, 1980.

[PUT 86] S.PUTHENPURA and N.K.SINHA
"Modified maximum likelihood method for the robust estimation of system parameters from very noisy data"
Automatica, vol.22, no.2, 231-235, 1986.

[TIK 82] M.L.TIKU and M.SINGH
"Robust statistics for testing mean vectors of multivariate distribution"
Commun. Statist. Theory. Meth., 11, 985-1001, 1982.

[ZEN 91] X.ZENG and C.VASSEUR
"Estimation of statistical parameters of a population from very noisy samples"
EURISCON'91: The european robotics and intelligence systems conference, june, 1991. Kanoni, Corfu, Greece.

[BAK 76] N.BAKHVALOV
Méthodes numériques
EDITION MIR.MOSCOU, 1976

[SIB 82] M.SIBONY et J.C.MARDON
Systèmes linéaires et non linéaires
HERMANN, 1982

[MAR 72] R.D.MARTIN
"Robust estimation of signal amplitude"
IEEE Trans. Inform. Theory, vol.IT-18, no.5, september 1972.

[MAR 75] R.D.MARTIN and C.J.MASRELIEZ
"Robust estimation via stochastic approximation"
IEEE Trans. Inform. Theory, vol.IT-21, no.3, may 1975.

[CAI 76] F.CAILLIEZ et J.G.PAGE
Introduction à l'analyse des données
SMASH, 1976

[DUD 79] P.DUDES et A.K.JAIN
"Validity studies in clustering methodologies"
Pattern Recognition, vol.11, pp.235-254, 1979

[GON 76] M.GONDRAN
"Valeurs propres et vecteurs propres en classification hiérarchique"
RAIRO Informatique Théorique, vol.10, n°3, mars 1976

[DEG 75] V.DEGOT et J.M.HUALDE
"De l'utilisation de la notion de clique (sous-graphe complet symétrique) en matière de typologie de population"
RAIRO, janvier 1975, V-1, p.5 à 18

[WAT 81] S.WATANABE
"Pattern Recognition as a quest for minimum entropy"
Pattern Recognition, vol.13, n°5, 1981

[BAL 67] G.H.BALL et D.J.HALL
"A clustering technique for summarizing multivariate data"
Behav. Sci. 12, 153-155, 1967

[KIT 88] J.KITTER et D.PAIRMAIN
"Optimality of reassignment rules in dynamic clustering"
Pattern Recognition, vol.21, n°2, pp.169-174, 1988

[ISM 89] M.A.ISMAIL et M.S.KAMEL
"Multidimensional data clustering utilizing Hybrid search strategies"
Pattern Recognition, vol.22, n°1, pp.75-89, 1989

[DUB 87] R.C.DUBES
"How many clusters are best? - an experiment"
Pattern Recognition, vol.20, n°6, pp.645-663, 1987

[DAV 79] D.L.DAVIES et D.W.BOULDIN
"A cluster separation measure"
IEEE Trans. PAMI, vol.PAMI-1, n°2, april, 1979

[LES 79] J.M.LESTER, H.A.WILLIAMS, B.A.WEINTRAUB et J.F.BRENNER
"Two graph searching techniques for boundary finding in white blood cell image"
Comput. Biol. Med. 8, 293-308, 1979

[VAN 81] L.VANDERHEYDT, F.DOM, A.OOSTERLINCK et H.V.BERGHE
"Two-dimensional shape decomposition using fuzzy subset theory applied to automated chromosome analysis"
Pattern Recognition 13, 147-157, 1981

[LIA 89] J.LIANG
"Intelligent splitting in the chromosome domain"
Pattern Recognition, vol.22, n°5, pp.519-532, 1989

PARTIE II

PILOTAGE DU PROCESSUS PAR OBSERVATION D'ECHANTILLONS

INTRODUCTION

Dans le système étudié, les échantillons temporels de la sortie constituent différentes classes caractérisées par des paramètres statistiques (vecteur moyenne et matrice de covariance). Pour modifier la structure courante du système, il s'avère nécessaire de définir une stratégie de pilotage afin de conduire les échantillons de la sortie du système vers la distribution désirée.

En général, la synthèse de cette stratégie de pilotage ne peut être faite que sous l'hypothèse que la structure caractéristique du système est complètement connue. Dans cette situation, un modèle mathématique est établi pour s'adapter à la connaissance du système et on cherche à améliorer les performances du système en introduisant des correcteurs du type avance, retard de phase ou P.I.D., etc. [BROGAN, 1974], [FARGEON, 1986].

Dans le cas où les paramètres du système sont inconnus, on cherche alors à adapter l'organe de commande de façon à ce que système se comporte comme le modèle de référence [NAJIM, 1982].

Si la connaissance du système n'est pas disponible et aucun modèle sur le système n'est à notre disposition, le système étudié peut être considéré comme un environnement inconnu. Dans cette situation, nous devons envisager une approche consistant à élaborer une stratégie de pilotage uniquement à partir de l'information fournie par l'observation de ce système. Le schéma de pilotage du système est donné dans la fig-2.1.

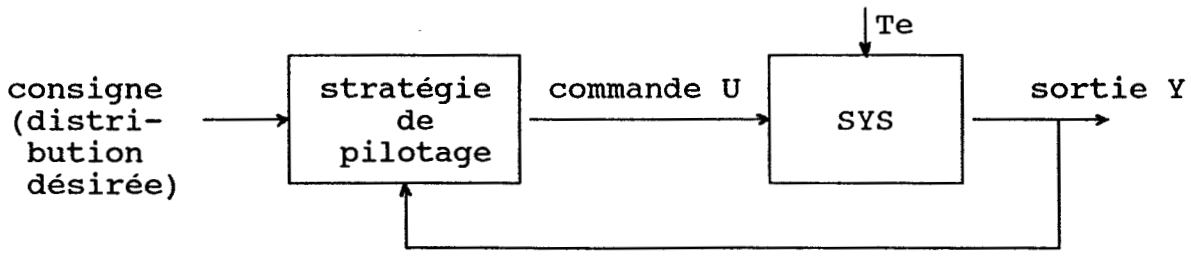


fig-2.1 pilotage du système en boucle fermée

SYS est le système étudié. A chaque instant d'échantillonnage (période=T_e), le système sort un nouvel échantillon Y. Le but de la stratégie de pilotage est d'adapter la distribution des échantillons de la sortie à la consigne désirée.

Comme dans les chapitres précédents, les échantillons de la sortie suivent des modèles de distribution normale, caractérisés par:

- vecteur moyenne: M

- matrice de covariance: S

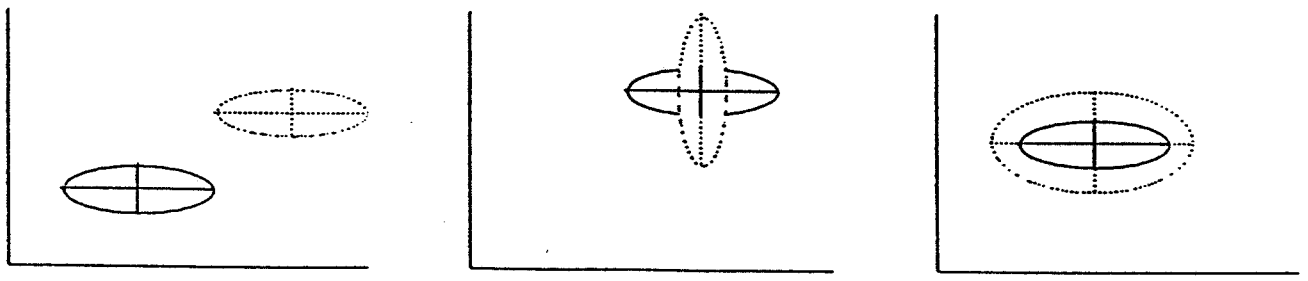
où S est définie par ses valeurs propres: λ_i (i=1, ..., d) et ses vecteurs propres: U_j (j=1, ..., d). (d est la dimension de l'espace d'observation).

La procédure de pilotage de ces paramètres conduit les échantillons d'une distribution vers l'autre. Elle contient 3 opérations principales (cf fig-2.2):

1) translation: conduite du vecteur moyenne initial des échantillons à la valeur désirée;

2) rotation: rotation des vecteurs propres dans les directions désirées;

3) dilatation: variation des valeurs propres vers les valeurs désirées.



translation rotation dilatation

fig-2.2 les 3 opérations du pilotage

Pour simplifier le pilotage, certaines hypothèses sont

imposées au système:

1) Les 3 opérations ci-dessus sont complètement indépendantes. Il n'existe pas d'interconnection entre ces 3 procédures de commande.

2) Le système est commandable et observable. Le vecteur moyenne, les valeurs propres et les vecteurs propres de la distribution des échantillons de la sortie peuvent être respectivement règlés par un vecteur de commande U pour conduire ces paramètres statistiques vers les consignes désirées.

3) La sortie Y doit être sensible aux variations des paramètres caractéristiques du système. De même, l'état du système doit être également sensible aux variations de commande U.

4) La sortie du système doit rester stable en régime stationnaire. La sortie est fixe si l'entrée reste invariante.

Dans les chapitres suivants, on propose deux approches pour réaliser cette stratégie de pilotage. Dans la première approche, un automate stochastique est élaboré et la stratégie de pilotage est générée par apprentissage de cet automate (Chapitre V). Dans la deuxième approche, un réseau neuronal artificiel est élaboré et la stratégie de pilotage du système est générée par apprentissage au sein du réseau (Chapitre VI). En conclusion, une comparaison entre ces deux approches est donnée.

CHAPITRE V
PILOTAGE PAR APPRENTISSAGE D'UN AUTOMATE STOCHASTIQUE

V.1 ARCHITECTURE DE LA STRATEGIE DE PILOTAGE

V.1.1 PILOTAGE DE TRANSLATION

Supposons qu'il existe r composantes de commande de translation (soit: $U=\{u_1, u_2, \ldots, u_r\}$). La dimension d'observation est d. A chaque période, la variation d'une composante de commande Δu_1 provoque les variations de sortie suivantes: $\Delta Y_{11}, \Delta Y_{12}, \ldots, \Delta Y_{1d}$. De la même manière, $\Delta u_2, \ldots, \Delta u_r$ provoquent respectivement d variations de sortie.

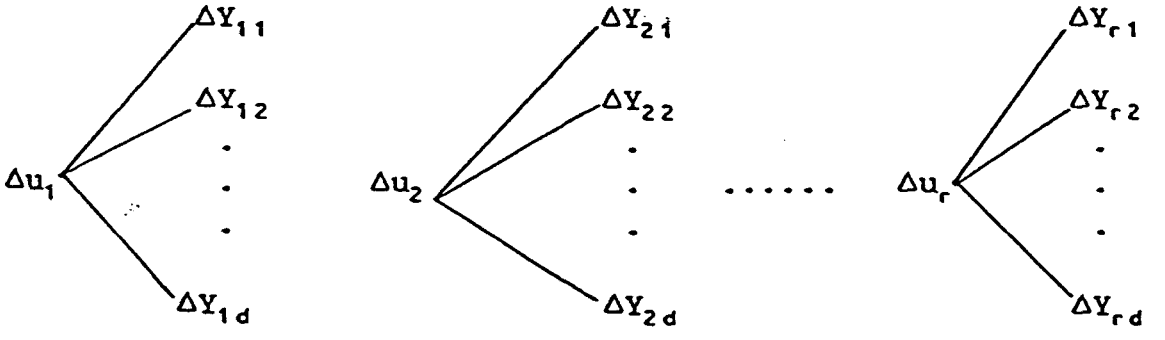


fig-2.3 relation entre la commande et la sortie

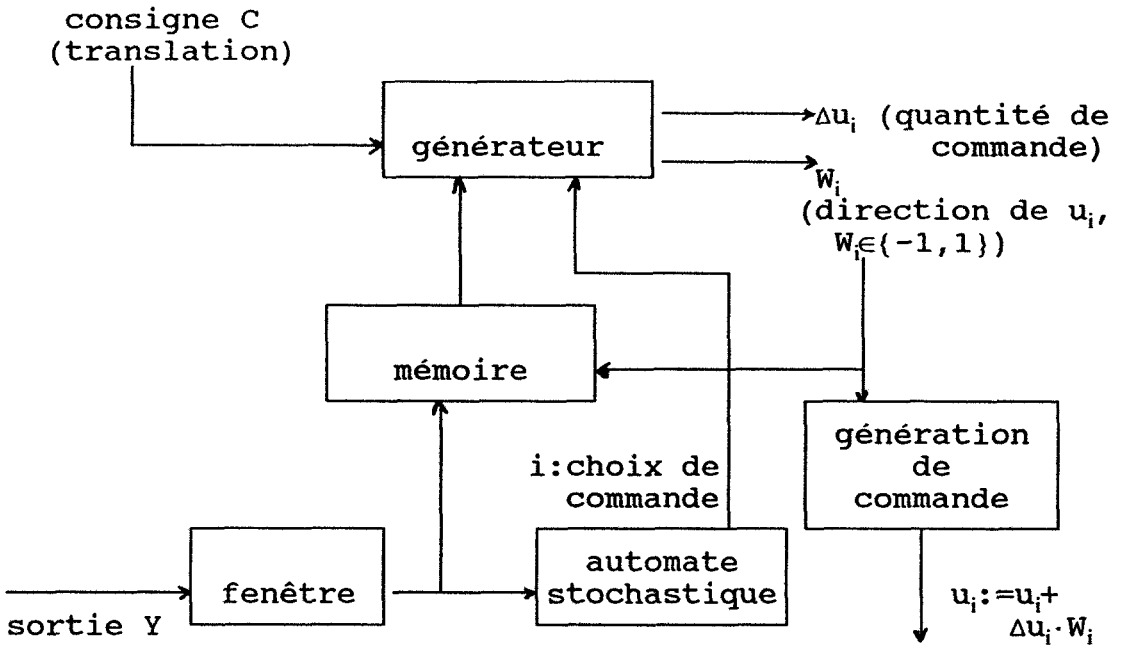


fig-2.4 stratégie de pilotage de translation

La stratégie de translation vise à amener le vecteur moyenne de la sortie du système le plus près possible du vecteur désiré.

V.1.2 PRINCIPE DE L'AUTOMATE STOCHASTIQUE

Un automate stochastique qui fonctionne dans un environnement inconnu est proposé comme modèle d'apprentissage. L'opération principale effectuée par cet automate est de renouveler les probabilités associées aux actions selon les réponses de l'environnement afin d'améliorer les performances selon des critères spécifiques. [NARENDRA and THATHACHAR, 1974]. Certains algorithmes de renouvellement des probabilités utilisent les informations d'interaction avec l'environnement pour générer des actions plus efficaces [THATHACHAR and SASTRY, 1985], [SIMHA and KUROSE, 1989]. Cette approche a été appliquée dans différents domaines. Dans [NAJIM, 1982], on présente une application de cet automate à la commande d'un canal d'irrigation. Une autre application de l'automate stochastique est la commande et le routage du traffic en téléphonie, effectuée par [NARENDRA, WRIGHT et MASON, 1977]. Dans cette approche, un automate est utilisé pour simuler le fonctionnent des réseau téléphoniques.

Un automate stochastique est un sextuplet (Y, Φ, α, P, A, G) où:

- Y représente l'entrée de l'automate;

- $\Phi=\{\phi_1, \phi_2, \ldots, \phi_s\}$ constitue l'ensemble des états internes;

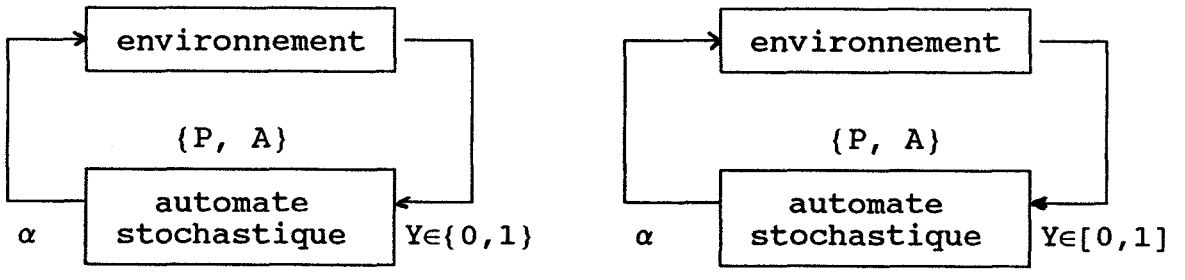
- $\alpha=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ avec $r \leq s$ est la sortie de l'automate.

P est un vecteur probabilité qui gouverne les transitions d'un état à l'autre à chaque période n. $P(n)=(p_1(n) \ldots p_s(n))^T$.

A est un algorithme ou schéma de renforcement qui engendre P(n+1) à partir de P(n), et G: $\Phi \rightarrow \alpha$ est la fonction de sortie.

Les sorties de l'automate constituent les entrées pour l'environnement et les réponses de ce dernier constituent les entrées de l'automate, qui influencent la mise à jour du vecteur probabilité P (cf fig-2.5).

Il est important de distinguer des modèles différents selon la nature de l'entrée de l'automate ou de la sortie de l'environnement. Si l'ensemble d'entrée est binaire, soit {0,1}, le modèle d'automate est un P-modèle. Si l'ensemble d'entrée est une collection finie de symboles différents, le modèle de l'automate est un Q-modèle. Si l'ensemble d'entrée est un segment [0,1], le modèle de l'automate est un S-modèle. Chaque type de modèle est adapté à des situations spécifiques.



(a) P-modèle (b) S-modèle

fig-2.5 automate stochastique et son environnement

V.1.3 ARCHITECTURE DE L'AUTOMATE PROPOSE

Dans l'étude présentée ci-dessous, l'automate stochastique est destiné à sélectionner une composante de commande parmi $\{u_1, u_2, \ldots, u_r\}$. Son domaine d'entrée est un espace euclidien de dimension d ($Y \in R^d$), ce qui est différent des modèles présentés ci-dessus.

Cet automate contient r états (r=s) associé chacun à une probabilité temporelle. Dans une période donnée, la sortie de l'automate correspond à l'état de probabilité d'occurrence maximale. Les entrées de l'automate sont les d composantes du vecteur de sortie. Les r probabilités temporelles évoluent selon les valeurs des entrées.

A chaque fois qu'une composante de commande se trouve dans un état favorable, la probabilité associée à cette composante doit être renforcée et l'état correspondant de l'automate doit devenir prioritaire. Dans ce cas, cette composante de commande peut être appliquée au système plusieurs fois jusqu'à ce que sa probabilité décroisse et qu'une autre composante de commande se trouve dans un état plus favorable.

V.2 ALGORITHME DE PILOTAGE PAR APPRENTISSAGE DE L'AUTOMATE

V.2.1 NOTATIONS PRINCIPALES DE L'ALGORITHME

L'évolution temporelle de l'automate stochastique se fait de la façon suivante:

Supposons qu'il existe r états dans l'automate. A l'instant n, les probabilités correspondant à ces états sont $P_1(n)$, $P_2(n)$, ..., $P_r(n)$.

La définition de $P_i(n)$ (i=1, ... r) doit être basée sur des invariants du système commandé. Par exemple, la direction de la sortie commandée par chaque composante de commande est invariante en régime stationnaire dans un système linéaire. La définition de $P_i(n)$ peut être établie en tenant compte de cette hypothèse.

Supposons que la sortie du système soit $Y=(y_1, y_2, ..., y_d)^T$ et que le vecteur désiré de la sortie $C=(c_1, c_2, ..., c_d)^T$. On propose le schéma de renforcement de l'automate (algorithme de pilotage) dans les sections suivantes.

V.2.2 INITIALISATION DE L'ALGORITHME DE PILOTAGE

On initialise à zéro les valeurs des composantes de commande:

$$u_1=u_2= \ ... \ =u_r=0$$

On définit pour toutes les composantes de commande un même accroissement uM, ce qui se traduit par les équations:

$\Delta u_1=\Delta u_2= \ ... \ =\Delta u_r=uM$ (quantité de commande)

$W_1=W_2= \ ... \ =W_r=1$ (direction de commande)

Dans la mesure où on ne dispose d'aucune information a priori sur le système, on ne sait pas initialement quelle est la composante de commande la plus performante pour conduire le système à la consigne. Les probabilités de l'automate sont alors définies de manière équiprobable:

soit: $P_1(0)=P_2(0)= \ ... \ =P_r(0)=1/r$

V.2.3 EVOLUTION DE L'ALGORITHME DANS LA PREMIERE PERIODE DE COMMANDE

A partir de la position initiale de la sortie Y_0, u_1, u_2, ..., u_r sont successivement appliquées au système: $u_i = W_i \cdot \Delta u_i$ (i=1, ..., r). Ceci signifie que l'on donne au vecteur U, successivement les valeurs: (uM, 0,..., 0), (uM, uM, 0, ..., 0), ..., (uM, uM, ..., uM).

On obtient alors les effets suivants:

La position Y_0 se déplace en Y_1 après l'application de u_1. Y_1 se déplace en Y_2 après l'application de u_2. ..... Y_{r-1} se déplace en Y_r après l'application de u_r. Ceci permet de définir les vecteurs de variations de la sortie de la manière suivante:

$$V_1 = Y_1 - Y_0,\ V_2 = Y_2 - Y_1,\ \ldots\ldots,\ V_r = Y_r - Y_{r-1}$$

Après application de tous les u_i (i=1,2,...,r), on calcule les probabilités associées en considérant les positions et les directions des vecteurs de sortie dans un espace de dimension d.

En notant $Z_1 = Y_r$ (position finale après l'application), on calcule respectivement les distances entre H_i et la consigne C, où H_i est le point d'intersection entre le vecteur V_i et la droite orthogonale à V_i passant par le point C.

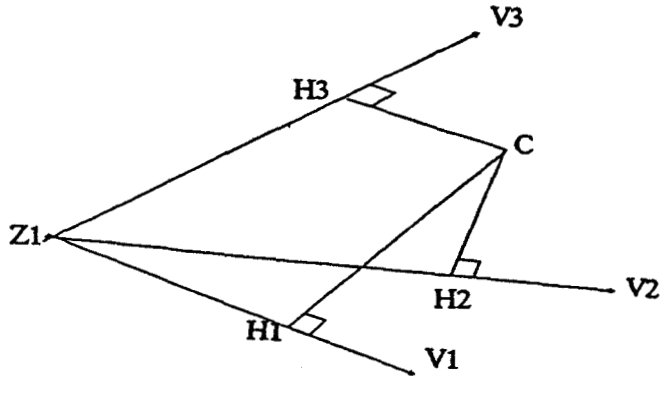


fig-2.6 distances entre H_i et la consigne C

On choisit alors la composante de commande correspondant à la distance la plus courte.

La distance pour le vecteur V_j est donnée ci-dessous:

$$d_j = \| C - H_j \| = \left(\sum_{i=0}^{d} (c_i - h_{ji})^2 \right)^{1/2} \qquad (2.1.1)$$

Sur la droite $Z_1 H_j$, on a:

$$\frac{h_{j1} - z_{11}}{v_{j1}} = \frac{h_{j2} - z_{12}}{v_{j2}} = \ldots = \frac{h_{jd} - z_{1d}}{v_{jd}} = f \qquad (2.1.2)$$

Par ailleur, $(H_j - C)^T V_j = 0$ (soit: V_j est orthogonal à la droite $H_j C$), on obtient:

$$v_{j1}(h_{j1} - c_1) + \ldots + v_{jd}(h_{jd} - c_d) = 0 \qquad (2.1.3)$$

A partir de (2.1.2) et (2.1.3), on obtient les équations suivantes:

$$f = \frac{\sum_{i=0}^{d} v_{ji}(c_i - z_{1i})}{\sum_{i=0}^{d} v_{ji}^2} \quad \text{et} \quad h_{ji} = z_{1i} + f \cdot v_{ji} \qquad (2.1.4)$$

Il vient: $d_j = \left(\sum_{i=1}^{d} (c_i - z_{ji} - f \cdot v_{ji})^2 \right)^{1/2}$ pour $\forall j \in \{1, 2 \ldots, r\}$

Après l'obtention de d_1, d_2, ..., d_r, les probabilités de l'automate $P_j(1)$ (j=1,2,...,r) sont définies par

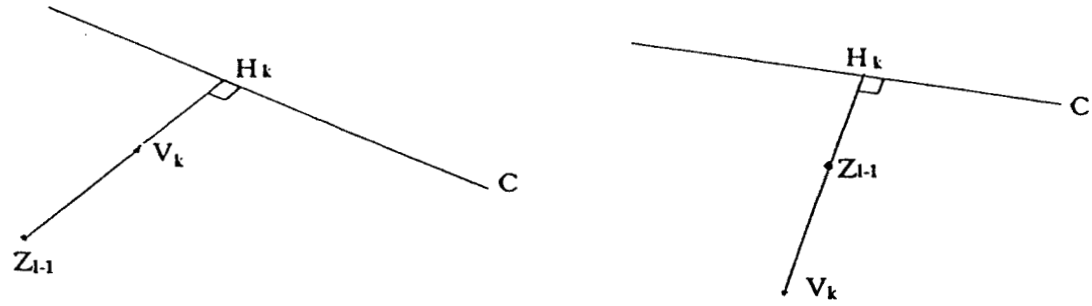
$$\begin{cases} P_1(1) = \rho_1(1/d_1) \\ P_2(1) = \rho_1(1/d_2) \\ \quad\vdots \\ \quad\vdots \\ P_r(1) = \rho_1(1/d_r) \end{cases} \qquad (2.1.5)$$

où ρ_1 est un coéfficient tel que $P_1(1) + P_2(1) + \ldots + P_r(1) = 1$

V.2.4 EVOLUTION DE L'ALGORITHME DANS LA PERIODE L

Dans la période l (l=2,3,...), on sélectionne la composante u_k telle que $P_k(l) > P_i(l)$ ($\forall i \in \{1,\ldots,r\}$ et $k \neq i$) et on obtient la position Z_l après l'application de u_k, c'est à dire de $U = U + W_k \cdot \Delta U$, avec $\Delta U = (0, \ldots, \Delta u_k, 0, \ldots, 0)$.

Pour l'application de u_k, la direction et la quantité de u_k sont définies de la façon suivante:



W_k ne change pas $\qquad\qquad$ $W_k := -W_k$

fig-2.7 définition du sens de u_k

Pour approcher la consigne, il est nécessaire que la direction de déplacement provoquée par u_k à la période l soit pointée vers H_k. Ceci se traduit par les conditions suivantes:

Si $V_k^I(Z_{l-1}H_k)>0$, alors la direction W_k ne change pas.

Si $V_k^I(Z_{l-1}H_k)<0$, alors $W_k := -W_k$.

Selon l'équation (2.1.4), on a: $h_{ki}-z_{1i}=f\cdot v_{ki}$ (2.1.6)

et les conditions ci-dessus s'écrivent:

$$V_k^I(H_k-Z_1)=\sum_i v_{ki}\cdot f\cdot v_{ki}=f\sum_i v_{ki}^2=\sum_i v_{ki}(c_i-z_{1i}) \qquad (2.1.7)$$

C'est à dire: si $\sum_i v_{ki}(c_i-z_{1i})>0$, alors W_k ne change pas;

si $\sum_i v_{ki}(c_i-z_{1i})<0$, alors $W_k := -W_k$.

La quantité de commande Δu_k est donnée de la manière suivante:

1) Si $\|V_k\|<\alpha\cdot\|CZ_{l-1}\|$, le déplacement provoqué par Δu_k est trop petit, alors $\Delta u_k(l):=\beta\cdot\Delta u_k(l-1)$ où $0<\alpha<1$ et $\beta>1$.

2) Si $\|V_k\| > \|CZ_{l-1}\|$, le déplacement provoqué par Δu_k est trop grand, alors $\Delta u_k(l) := \frac{1}{\beta} \cdot \Delta u_k(l-1)$.

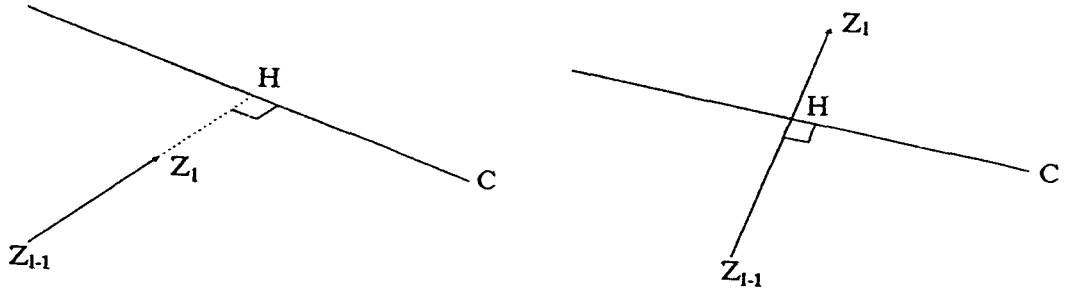
3) Dans les autres cas, Δu_k ne change pas ($\Delta u_k(l) = \Delta u_k(l-1)$).

Dans la simulation, on choisit $\alpha=0.1$, $\beta=2$ et la valeur initiale des Δu_i ($i=1, 2, \ldots, r$) est $uM = 0.1 \cdot \|Y_0 C\|$.

Finalement, on obtient $u_k(l) := u_{k-1}(l-1) + W_k \cdot \Delta u_k(l)$.

Après l'application de u_k, on affecte le vecteur d'effet V_k la nouvelle valeur $V_k := Z_l - Z_{l-1}$.

On obtient également le nouveau point d'intersection H de V_k avec sa droite orthogonale passant par C.



même coté différents cotés

fig-2.8 positions relatives entre Z_{l-1} et Z_l

Avant de définir les nouvelles probabilités temporelles de l'automate stochastique, il faut discuter de la position relative entre Z_{l-1} et Z_l par rapport à la ligne CH.

Ceci se traduit par les équations suivantes:

Si $(HZ_l)^T (HZ_{l-1}) > 0$, Z_{l-1}, Z_l se trouvent du même coté.
Si $(HZ_{l-1})^T (HZ_l) < 0$, Z_{l-1}, Z_l se trouvent de part et d'autre de HC.

$$(HZ_l)^T (HZ_{l-1}) = \sum_i (z_{l\,i} - h_i)(z_{l-1,\,i} - h_i)$$

$$= \sum_i (z_{l\,i} - z_{l-1,\,i} - fv_{k\,i})(z_{l-1,\,i} - z_{l-1,\,i} - fv_{k\,i})$$

$$=f(f-1)\sum_i v_{ki}^2=(\sum_i v_{ki}^2)\frac{\sum_i v_{ki}(c_i-z_{l-1,i})}{\sum_i v_{ki}^2}\cdot\frac{\sum_i v_{ki}(c_i-z_{li})}{\sum_i v_{ki}^2} \qquad (2.1.8)$$

Etant donné que $\sum_i v_{ki}^2>0$, on obtient les résultats suivants:

Si $\sum_i v_{ki}(c_i-z_{l-1,i})\cdot\sum_i v_{ki}(c_i-z_{li})>0$, alors Z_{l-1} et Z_l se trouvent du même coté de HC.

Si $\sum_i v_{ki}(c_i-z_{l-1,i})\cdot\sum_i v_{ki}(c_i-z_{li})<0$, alors Z_{l-1} et Z_l se trouvent de part et d'autre de HC.

On obtient alors les nouvelles probabilités suivantes:

Si Z_{l-1}, Z_l se trouvent du même coté de CH, ceci signifie que l'évolution est correcte et la sélection de u_k sera donc maintenue. Les probabilités sont alors définies par

$$\begin{cases} P_k(1)=1 \\ P_i(1)=0 \qquad \text{pour } \forall i\neq k \text{ et } i\in\{1,\ldots,r\} \end{cases} \qquad (2.1.9)$$

Si Z_{l-1}, Z_l se trouvent de part et d'autre de CH, la composante u_k est remplacée par une autre en fonction des probabilités suivantes:

$$\begin{cases} P_k(1)=0 \\ P_i(1)=\rho_l/d_i \qquad \text{pour } \forall i\neq k \text{ et } i\in\{1,\ldots,r\} \end{cases} \qquad (2.1.10)$$

où $d_i=\sum_j (c_j-z_{lj}+f\cdot v_{ij}^2)^{1/2}$ et $f=\dfrac{\sum_j v_{ij}(c_j-z_{lj})}{\sum_j v_{ij}^2}$

$V_i=(v_{i1},\ \ldots,\ v_{id})$ étant le vecteur d'effet après l'applica-

tion de u_i ($i \in \{1,\ldots,r\}$).

ρ_l est un coéfficient tel $P_1(1)+\ldots+P_r(1)=1$.

La procédure V.2.4 se répète jusqu'à la vérification de la condition suivante:

$\|Y-C\| < \varepsilon$ ε étant l'écart désiré. (2.1.11)

V.3 APPLICATION AUX SYSTEMES LINEAIRES

V.3.1 RAPPEL SUR LES SYSTEMES LINEAIRES A REGIME STATIONNAIRE

Dans un système linéaire et stationnaire, les équations d'état et de sortie peuvent s'écrire de la manière suivante:

$$\begin{cases} X(k+1)=AX(k)+BU(k) \\ Y(k)=CX(k)+DU(k) \end{cases} \qquad (2.1.12)$$

où X est vecteur d'état et Y vecteur de sortie

Les valeurs stationnaires de X(k) et Y(k) sont obtenues d'après (2.1.12) par:

$$X=(I-A)^{-1}BU \qquad (2.1.13)$$

$$Y=C(I-A)^{-1}BU+DU=GU \qquad (2.1.14)$$

où $G=C(I-A)^{-1}B+D$ est une matrice d×r constante.

Evidemment, le système est stable si et seulement si toutes les valeurs propres de A: a_i satisfont la condition suivante: $-1<a_i<1$ ($i\in\{1,2,\ldots,\}$) [BROGAN, 1974].

Selon l'algorithme défini précédemment, à chaque période, il n'y a qu'une seule composante de commande u_i qui varie. C.a.d: $\Delta u_l \neq 0$ et $\Delta u_i = 0$ pour $\forall i \neq l$, $i\in\{1,\ldots,r\}$.

$$\text{d'où: } \Delta Y=G\cdot\Delta U=(g_{1l},g_{2l},\ldots,g_{dl})\Delta u_l \qquad (2.1.15)$$

Par conséquent, le vecteur d'effet V_i de la sortie associé à chaque composante de commande est constant dans le cas linéaire.

V.3.2 TRAJECTOIRE THEORIQUE DE SORTIE DU SYSTEME

En général, la stratégie de commande présentée ci-dessus peut garantir la convergence de la trajectoire de sortie vers la consigne.

Dans l'espace multi-dimensionnel du vecteur de sortie, la procédure de commande s'effectue dans un polyèdre dont le sommet

est la consigne C. Chaque composante de commande est appliquée jusqu'à ce que son vecteur d'effet V_i traverse l'hyper-plan orthogonal à ce vecteur et passant par le point C. Dans le cas linéaire, ces hyper-plans sont fixes et définissent le polyèdre. Selon l'algorithme proposé ci-dessus, la trajectoire de sortie converge vers la consigne si on choisit une quantité de commande initiale appropriée.

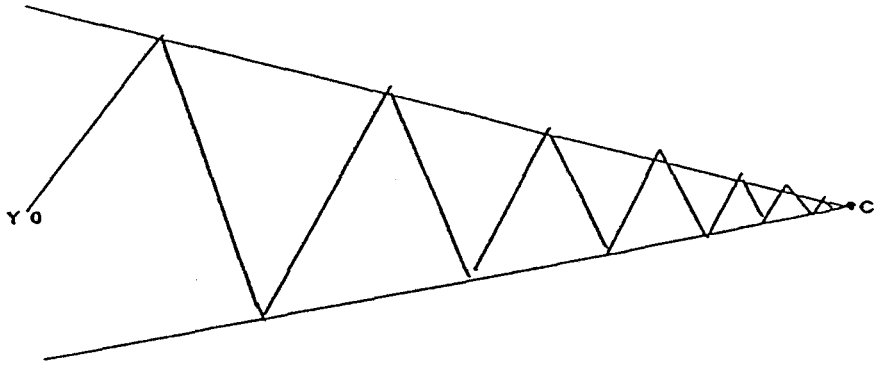


fig-2.9 trajectoire théorique de la sortie dans le cas linéaire

Dans un système non-linéaire, la convergence de l'algorithme n'est pas garantie. Dans ce cas en effet, on doit soit redéfinir l'automate stochastique à partir de nouvelles hypothèses, soit transformer le système courant en système linéaire (linéarisation).

V.4 PILOTAGE DES VALEURS PROPRES ET DES VECTEURS PROPRES

V.4.1 PRINCIPE DU PILOTAGE DES VALEURS PROPRES

Le nuage composé des échantillons de sortie du système est une super-ellipse dans l'hypothèse d'une distribution normale.

Supposons que $L=(\lambda_1,\lambda_2,\ldots,\lambda_d)^T$ soient les valeurs propres du nuage. Evidemment, $\lambda_i>0$ pour $\forall i\in\{1, 2, \ldots, d\}$. Dans la pratique, lorqu'une valeur propre λ_i s'approche de 0, elle devient quasi-invariante car elle n'est plus commandable du fait des perturbations de sortie.

L'algorithme de translation, présenté dans V.2, est basé sur l'hypothèse qu'aucune contrainte n'existe sur les valeurs des composantes de Y ($-\infty<y_i<\infty$ pour $\forall i\in\{1, \ldots, d\}$). Ainsi, pour appliquer cet algorithme dans la dilatation, il est nécessaire de transformer les valeurs bornées des composantes de L en valeurs non-bornées.

D'après ce principe, on affecte à la sortie du système un module de transformation des valeurs propres selon une loi logarithmique. Dans le nouveau système (U-Z) (cf fig-2.10), la sortie devient commandable et on peut appliquer la stratégie de commande de translation à ce système.

En imposant $Z=(z_1,z_2,\ldots,z_d)^T=\ln(L)$ à la sortie du système SYS, on obtient la relation $-\infty<z_i<+\infty$.

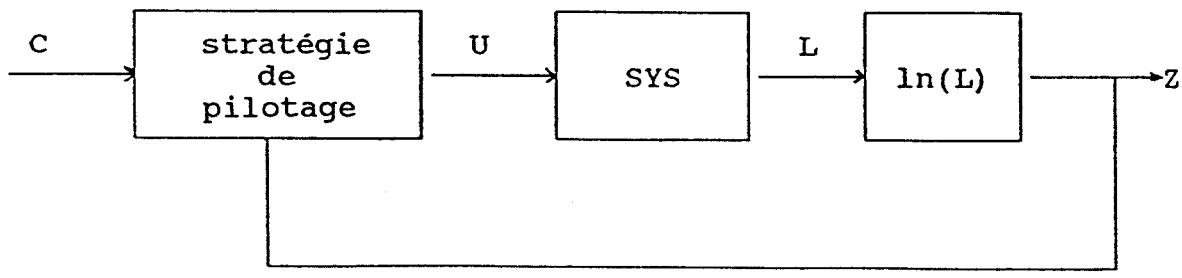


fig-2.10 architecture de commande pour les valeurs propres

Dans la simulation, on suppose que le système (U-Z) est linéaire et on utilise la stratégie de pilotage ci-dessus pour conduire la sortie du système vers les valeurs propres désirées.

V.4.2 PRINCIPE DU PILOTAGE DES VECTEURS PROPRES

Selon le même principe, on effectue les transformations pour la commande des vecteurs propres (rotation).

Supposons que les vecteurs propres du nuage de la sortie soient V_1, V_2, ..., V_d, où $V_i=(v_{i1}, v_{v2}, ..., v_{id})^T$ pour $\forall i\in\{1,2,...,d\}$.

On obtient une matrice orthonormée $V=[V_1,V_2,...,V_d]$

$$\text{soit: } V=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1d} \\ v_{21} & v_{22} & \cdots & v_{2d} \\ & \cdots & \cdots & \\ v_{d1} & v_{d2} & \cdots & v_{dd} \end{bmatrix} \qquad (2.1.16)$$

Les éléments de la matrice ne sont pas indépendants et ils satisfont aux conditions suivantes:

$$\sum_{k=1}^{d} v_{ki}^2=1 \text{ et } \sum_{k=1}^{d} v_{ki}v_{kj}=0 \quad \text{pour } \forall i,j\in\{1,2,...,d\} \text{ et } i\neq j \qquad (2.1.17)$$

Sous ces contraintes, il n'y a que $d(d-1)/2$ éléments indépendants dans V; c.a.d: les v_{ij} $(i+j\leqslant d,\ i,j=1,...,d)$. Les autres éléments peuvent être obtenus à partir des équations (2.1.17).

Les éléments v_{ij} $(i+j\leqslant d)$ doivent satisfaire aux inégalités suivantes:

$$\begin{cases} v_{11}^2+v_{21}^2+...+v_{d-1,1}^2\leqslant 1 \\ v_{12}^2+v_{22}^2+...+v_{d-2,2}^2\leqslant 1 \\ \qquad ..... \\ v_{1,d-1}^2\leqslant 1 \end{cases} \qquad (2.1.18)$$

Pour l'inégalité $v_{1i}^2+v_{2i}^2+...+v_{d-i,i}^2\leqslant 1$ où $i\in\{1,2,...,d-1\}$, on effectue la transformation suivante:

$$\begin{cases} v_{1i}=r_i\cdot\cos\theta_{i,d-i-1}\cdot\ldots\cdot\cos\theta_{i1} \\ v_{2i}=r_i\cdot\cos\theta_{i,d-i-1}\cdot\ldots\cdot\cos\theta_{i2}\cdot\sin\theta_{i1} \\ \qquad\qquad\cdots\cdots \\ v_{d-i-1,i}=r_i\cdot\cos\theta_{i,d-i-1}\cdot\sin\theta_{i,d-i-2} \\ v_{d-i,i}=r_i\cdot\sin\theta_{i,d-i-1} \end{cases} \qquad (2.1.19)$$

où $0<r_i<1$ pour $i\in\{1,\ldots,d-1\}$, $0<\theta_{ij}<\pi$ pour $\forall j\in\{2,3,\ldots,d-i-1\}$ et $0<\theta_{i1}<2\pi$

On obtient les nouvelles variables de sortie suivantes à partir des $\{v_{ij}\}$ $(i+j\leqslant d)$:

$$r_1,\theta_{11},\theta_{12},\ldots,\theta_{1,d-2};\ r_2,\theta_{21},\theta_{22},\ldots,\theta_{2,d-3};\ \ldots\ r_{d-1}.$$

Les $\{r_i\}$ et $\{\theta_{ij}\}$ étant bornés, on effectue la transformation supplémentaire suivante:

$$\begin{cases} z_{1i}=\ln(r_i)-\ln(1-r_i) \\ z_{2i}=\ln(\theta_{i1})-\ln(2\pi-\theta_{i1}) \\ z_{ji}=\ln(\theta_{i,j-1}+\pi)-\ln(\pi-\theta_{i,j-1}) \end{cases} \qquad (2.1.20)$$

Evidemment, $-\infty<z_{ji}<+\infty$. La nouvelle sortie est alors non-bornée et nous supposons que le système (U-Z) est linéaire. La dimension de l'espace Z est $d(d-1)/2$. Dans ces conditions, le système peut être commandé par l'algorithme utilisé pour les vecteurs moyennes et les valeurs propres.

V.5 FILTRAGE DES ECHANTILLONS PAR FENETRAGE

Dans la stratégie de commande, les échantillons de la sortie sont les seules informations disponibles sur le système. La performance de la commande est directement influencée par les perturbations de ces échantillons. Pour atténuer l'effet de ces perturbations, on impose un filtre aux échantillons de sortie. Les valeurs filtrées sont entrées dans le régulateur de commande (cf fig-2.11). Le filtre proposé utilise la méthode de fenêtrage (cf Section I.3).

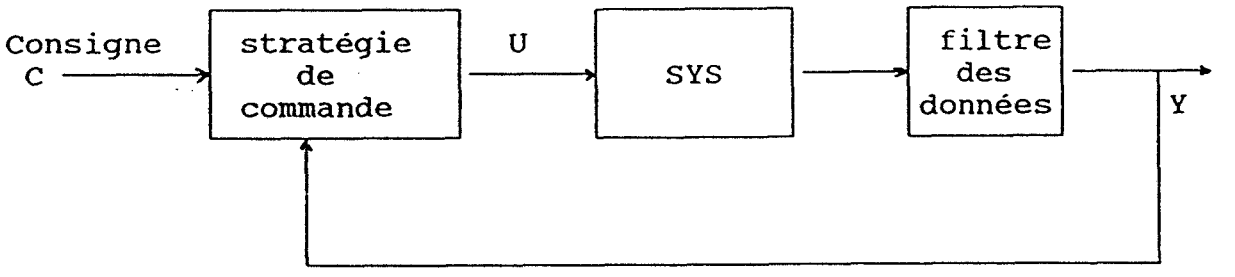


fig-2.11 schéma du système de commande avec filtre

A chaque période de commande, il existe deux types d'échantillons:

1) Les échantillons du régime transitoire concernant le changement de composantes de commande au début de la période.

2) Les échantillons du régime stationnaire.

Evidemment, la sortie réelle du système, qui correspond à l'entrée courante U, doit être estimée à partir des échantillons en régime stationnaire. Selon ce principe, on définit deux fenêtres de tailles n_1 et n_2 dans le filtre (cf fig-2.12).

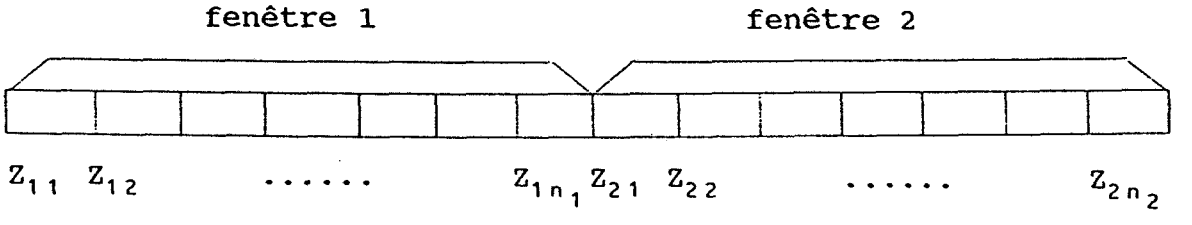


fig-2.12 fenêtres dans le filtre

La fenêtre 1 représente les n_1 premiers échantillons du régime transitoire, qui sont éliminés par le filtre.

La fenêtre 2 représente les n_2 derniers échantillons du

régime stationnaire. Dans la translation, on prend la moyenne des échantillons de la fenêtre 2 comme valeur filtrée.

$$\text{On obtient: } Y = \frac{1}{n_2}\sum_{i=1}^{n_2} Z_{2i} \qquad (2.1.21)$$

Si chaque Z_{2i} respecte la loi normale (c.a.d: $Z_{2i} \sim \mathbb{N}(M,S)$), alors on peut déduire $Y \sim \mathbb{N}(M,S/n_2)$. La variance de chaque composante de Y est divisée par n_2, c'est à dire, les perturbations sur les échantillons stationnaires sont divisées par n_2.

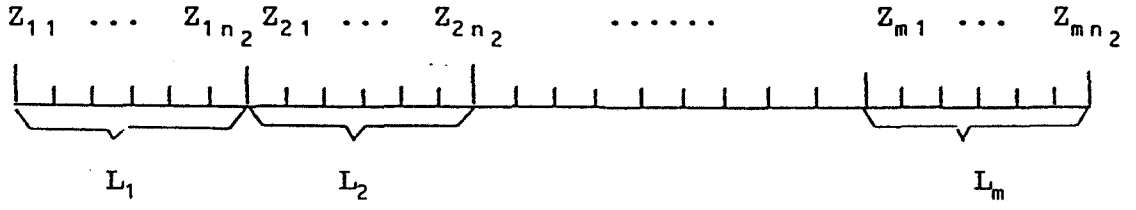


fig-2.13 fenêtres pour la commande des valeurs propres

Dans la commande des valeurs propres, on prend une fenêtre 2 d'échantillons (Z_{11}, ..., Z_{mn_2}) en régime stationnaire à chaque période de commande.

Cette fenêtre est divisée en m sous fenêtres et on effectue une estimation des valeurs propres pour chaque sous fenêtre (Z_{i1}, ..., Z_{in_2}) (i=1,..., m). Les valeurs propres sont estimées m fois par L_1, L_2, ..., L_m. La moyenne $L = \frac{1}{m}\sum_{i=1}^{m} L_i$ est alors considérée comme le vecteur des valeurs propres actuelles et traitée par le module de transformation.

Nous appliquons la même méthode dans la commande des vecteurs propres.

V.6 SIMULATION

V.6.1 PRINCIPE DE LA SIMULATION

Le but de la simulation est de tester la convergence et les performances de la stratégie de pilotage au cours des 3 opérations: translation, rotation et dilatation.

Pour la translation, le schéma de travail à l'intérieur d'une période de commande se présente de la façon suivante:

Etape 1:

Calcul d'une commande selon la stratégie présentée ci-dessus, comprenant la sélection de la composante de commande, la détermination de la quantité et de la direction de cette composante.

Etape 2:

Application de la commande au système. Extraction d'un sous ensemble (fenêtre 2) d'échantillons stationnaires par élimination des échantillons transitoires (fenêtre 1).

Etape 3:

Filtrage des échantillons stationnaires (fenêtre 2) et calcul du vecteur moyenne.

Etape 4:

Entrée du vecteur précédent dans le générateur de commande et retour à l'étape 1.

Les procédures de rotation et de dilatation sont similaires à la translation. La différence se trouve dans l'étape 4 où un module de transformation est introduit entre la sortie de la fenêtre 2 et l'entrée du générateur.

V.6.2 RESULTATS DE SIMULATION

Les systèmes choisis pour la simulation sont linéaires et stables. La simulation en translation s'effectue avec $n_1=n_2=5$ où n_1 est la longueur de la fenêtre 1 et n_2 la longueur de la fenêtre 2. Nous donnons ci-dessous 3 exemples de simulation dans différentes dimensions.

simulation de translation	d	r	Consigne	Y(fin)	U(fin)	Y(début)	nb
EX1	1	2	20	19.86	6.16 0.03	-0.05	50
EX2	2	2	10 8	9.92 7.92	-21.4 34.9	-0.03 -0.08	75
EX3	3	3	10 12 14	9.92 11.9 14.0	-19.0 34.8 7.44	-0.09 0.34 -0.02	65

où Y(début) est la sortie initiale et Y(fin), U(fin) sont respectivement la sortie et l'entrée finales; nb est le nombre de périodes de commande.

tableau 2.1 Simulation de translation

La trajectoire correspondant à la translation de l'EX2 est donnée ci-dessous:

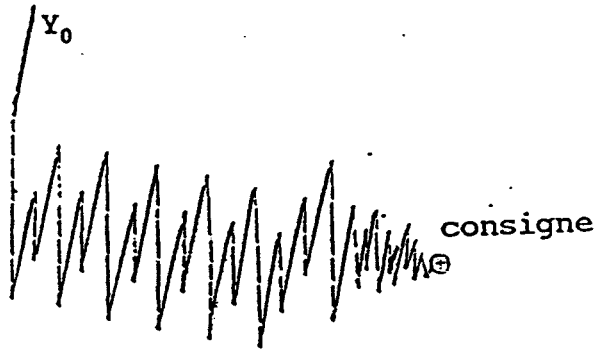


fig-2.14 trajectoire de translation (d=2 et r=2)

Pour la simulation en rotation et en dilatation, on choisit n_1=15 et n_2=90. La fenêtre 2 est divisée en 6 groupes de 15 échantillons pour l'estimation des valeurs propres ou vecteurs propres. Les résultats de simulation pour le système EX2 (d=2) sont donnés ci-dessous dans les espaces transformés.

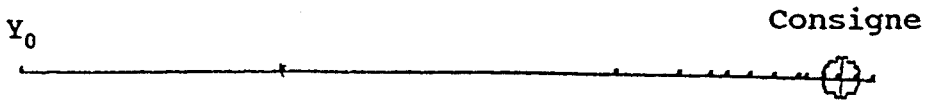


fig-2.15 trajectoire en rotation dans l'espace Z (U-V-Z) où la dimension de sortie est d(d-1)/2=1, la dimension de commande r=2

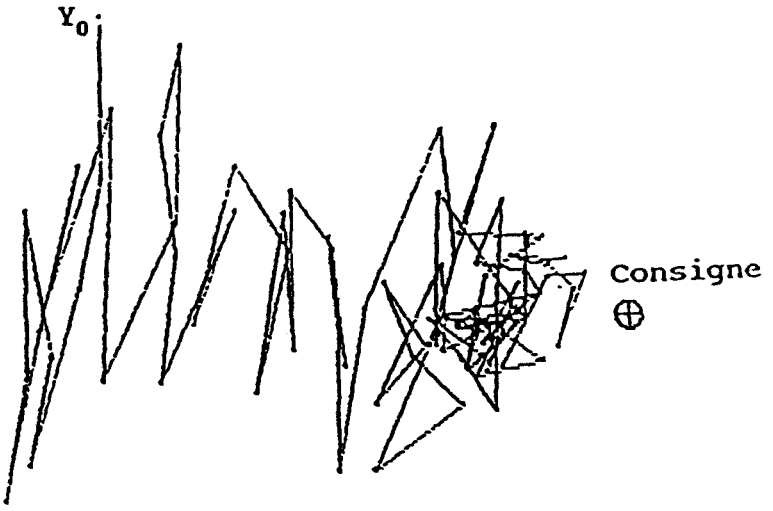


fig-2.16 trajectoire graphique en dilatation dans l'espace Z (U-L-Z) où les dimensions de sortie et de commande sont 2

Selon les exemples de simulation, la stratégie de pilotage peut garantir la convergence vers la forme désirée pour un système linéaire. Les résultats de simulation se sont également montrés probants pour d'autres exemples où les dimensions de sortie et d'entrée sont plus élevées (d=4,r=4; etc).

V.6.3 REMARQUES FINALES

L'étude effectuée permet de conduire en temps réel un système vers un état désiré sans modéliser la structure de ce système. En réalité, les systèmes sont parfois difficiles à modéliser et les paramètres caractéristiques correspondants ne sont pas toujours accessibles. Dans ce cas, l'observation des échantillons de la sortie apparaît comme une ressource essentielle permettant l'analyse et la conduite de ces systèmes.

Les fenêtres d'observation des échantillons de la sortie du système permettent d'obtenir une image immédiate et précise sur la situation du système. Grâce aux données fournies à chaque fenêtre, les composantes de commande peuvent être déterminées selon la stratégie de pilotage définie dans ce chapitre et être appliquées au système à la période suivante.

Les applications montrent de bonnes performances pour les systèmes linéaires et stationnaires. Théoriquement, cette stratégie peut être appliquée pour tous les types de systèmes si elle respecte les proprietés associées à l'architecture de

ces systèmes. Toutefois, ces proprietés sont parfois difficiles à mettre en évidence. Dans la pratique, un manque de connaissance sur les propriétés du système ne permet pas d'appliquer l'algorithme associé à l'automate stochastique. Malgré ces inconvenients, la stratégie de pilotage à partir des données de sortie est une méthode intéressante dans la conduite des systèmes pour lesquels aucun modèle mathématique n'est disponible a priori.

CHAPITRE VI
PILOTAGE PAR APPRENTISSAGE DES RESEAUX NEURONAUX

VI.1 RESEAUX NEURONAUX ARTIFICIELS DANS LE PILOTAGE

VI.1.1 INTRODUCTION DES RESEAU NEURONAUX

Ce chapitre propose une méthode de pilotage utilisant une architecture à réseau neuronal. Un réseau neuronal artificiel est un ensemble d'éléments interconnectés simulant le fonctionnement du cerveau humain. Par cette architecture, il est possible d'effectuer des essais permettant d'explorer les propriétés du système étudié. Dans ce sens, le réseau est capable d'apprendre. L'apprentissage d'un fait s'effectue par renouvellement des connexions entre les neurones, sur la base d'exemples présentés au réseau. On aboutit ainsi à une auto-organisation du réseau se traduisant par la mémorisation du fait.

Compte tenu de leurs propriétés, les réseaux neuronaux sont souvent utilisés dans des environnements inconnus pour effectuer des essais d'actions-effets sur un système, par apprentissage.

Un exemple bien connu est celui de la commande du pendule inverse. Dans ce problème, on construit une fonction permettant la sélection des actions d'entrée du système parmi toutes celles possible. La mise en oeuvre de réseaux neuronaux par essais d'actions-effets sur le système permet d'améliorer la qualité de la sélection opérée par cette fonction. Au cours de cette mise en oeuvre, les poids affectés aux réseaux sont modifiés, [ANDERSON, 1989], [BARTO, SUTTON et ANDERSON, 1983].

Dans beaucoup de modèles de réseaux neuronaux, un algorithme de rétro-propagation est utilisé pour modifier les poids de connexion. Après renouvellement des poids par apprentissage, les réseaux peuvent mieux s'adapter à l'environnement courant et à l'objectif, [HINTON, 1989], [PSALTIS, SIDERIS et YAMAMURA, 1988], [DAVALO and NAIM, 1989]. Dans la recherche de [BAVARIAN, 1988], la modification des

poids des réseaux neuronaux est basée sur la minimisation d'une fonction d'énergie.

De manière générale, un réseau neuronal artificiel, comme l'automate stochastique présenté dans le Chapitre V, effectue la recherche des entrées de l'environnement les mieux adaptées aux sorties désirées, selon un renouvellement itératif des poids de connexion. Actuellement, les algorithmes des réseaux neuronaux sont appliqués dans les domaines de la reconnaissance adaptative des formes, de la reconnaissance de la parole en temps réel, de l'identification en temps réel et de la commande des systèmes difficilement modélisables.

VI.1.2 PRINCIPE DE L'ALGORITHME PROPOSE

L'objectif de l'algorithme proposé dans ce chapitre est de conduire la sortie du système multi-variables (SYS) vers une consigne désirée sans faire appel à un modèle mathématique. Ce pilotage est réalisée uniquement à partir de l'observation des échantillons de la sortie du système que l'on suppose observable. Un réseau neuronal à trois couches (deux étages) est construit pour élaborer les entrées évolutives du système conduisant à la sortie désirée. Pendant toute la procédure de pilotage, les poids de connexion du réseau sont renouvelés en tenant compte des résultats des actions-effets sur le système. Initialement, la dynamique du système est inconnue. Au cours du pilotage, l'environnement du réseau est capable d'évaluer les conséquences de l'ensemble des actions effectuées par toutes les couches du réseau adaptif qui acquiert la connaissance de l'environnement en tenant compte de rétro-actions qui dépendent généralement des actions effectuées aux périodes précédentes. Dans ces conditions, les performances du système évaluées à partir d'un critère donné tendent à s'améliorer.

Le pilotage du système qui est proposée s'effectue dans un espace euclidien multi-dimensionnel et les poids de connexion sont renouvelés d'après l'observation des positions et directions de déplacement du vecteur de sortie. Trois procédures sont utilisées au cours du pilotage:

- une procédure d'élimination des perturbations de l'environnement pour l'obtention d'une image précise stationnaire sur la sortie du système. Cette procédure utilise les deux fenêtres d'échantillons définies la Section V.5.

- le premier étage du réseau neuronal permet d'évaluer les incréments de quantité et la direction appliqués au vecteur de commande pour la période suivante. Le calcul s'effectue à partir des signaux d'évaluation de la sortie, de la commande actuelle et des données mémorisées.

- le seconde étage du réseau détermine le sens du vecteur des incréments de commande à partir des signaux d'évaluation des périodes précédentes.

Dans les Sections VI.2, on rappelle le modèle du système à piloter et l'architecture du réseau qui fonctionne comme un générateur de commande. Dans la Section VI.3, l'algorithme de renouvellement des poids de connexion du premier étage est énoncé en détail. Dans la Section VI.4, l'algorithme de détermination du sens de commande par le deuxième étage du réseau est présenté. La convergence de l'algorithme et les résultats de simulation sont discutés dans la Section VI.5.

VI.2 ARCHITECTURE DE PILOTAGE A RESEAU NEURONAL

VI.2.1 MODELE DE LA STRATEGIE DE PILOTAGE

Nous rappelons que l'entrée et la sortie du système SYS sont multi-dimensionnelles de dimensions respectives r et d.

Soit $U=(u_1, u_2, \ldots, u_r)^T$ le vecteur d'entrée

et $Y=(y_1, \ldots, y_d)^T$ le vecteur de sortie

La variation Δu_1 de la composante u_1 de la commande provoque des variations sur la sortie notées: $\Delta Y_{11}, \Delta Y_{12}, \ldots, \Delta Y_{1d}$. De même pour $\Delta u_2, \ldots, \Delta u_r$ (cf fig-2.2).

Dans ce chapitre, on utilise un réseau neuronal comme la stratégie de pilotage du système SYS. Le schéma de commande est alors le suivant.

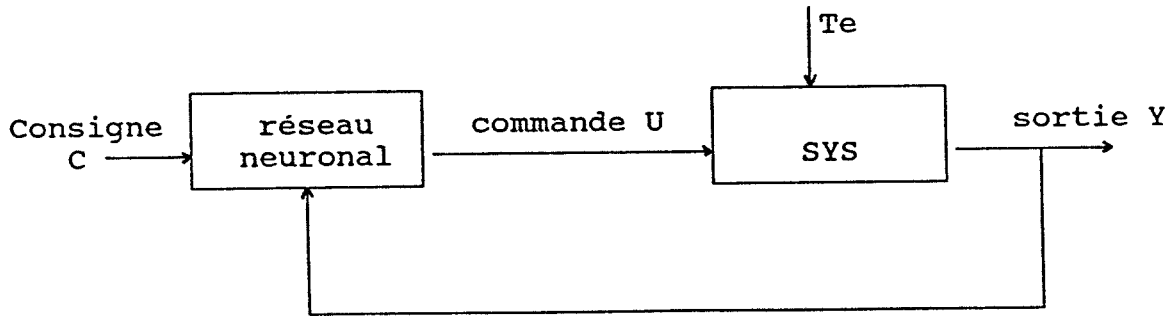


fig-2.17 pilotage du système en boucle fermée

Chaque échantillon Y est caractérisé par les paramètres M et S.

Pour simplifier, nous n'étudierons dans les sections suivantes que le cas de la conduite de M vers le vecteur de consigne C (translation). Au cours de cette procédure, la matrice de covariance S est supposée fixée. Dans ce sens, elle affecte la précision d'estimation de M.

VI.2.2 MODELE DU RESEAU NEURONAL

Le réseau neuronal utilisé possède une structure à trois couches (deux étages) pour réaliser le pilotage (fig-2.18). Le premier étage est destiné à déterminer la direction (sans sens) et la quantité de commande. Le deuxième étage détermine le sens du vecteur de commande. Les éléments de la couche 1 et de la couche 3 constituent respectivement les entrées et les sorties du réseau. La couche 2, appelée couche cachée, contient des éléments intermédiaires entre la couche 1 et la couche 3.

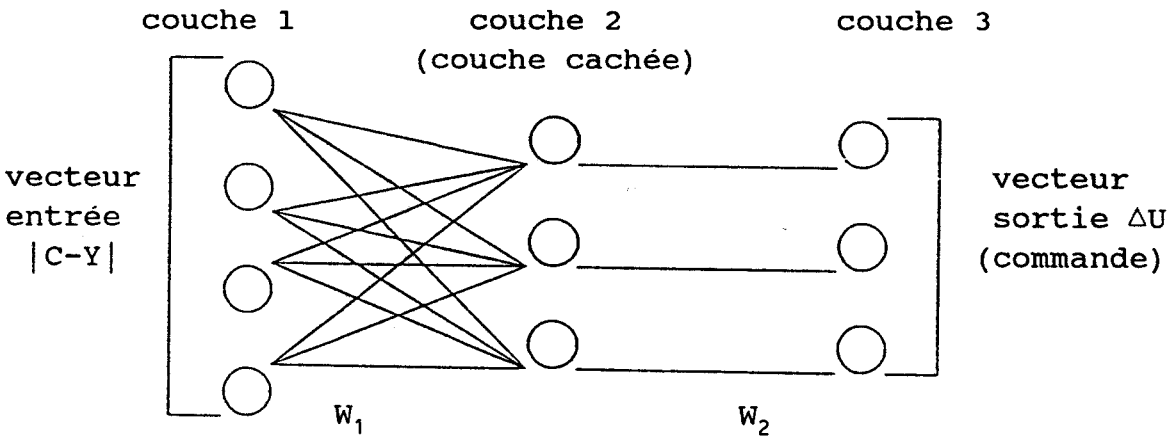


fig-2.18 architecture de pilotage à réseau neuronal

Dans ces conditions, au début de chaque période k+1, un vecteur de commande U est appliqué au système avec

$$U(k+1)=U(k)+\Delta U(k+1) \qquad (2.2.1)$$

où $\Delta U(k+1)$ est le vecteur des incréments de commande entre les périodes k+1 et k. $\Delta U(k+1)$ est déterminé à la période k.

Si on note $W_1(k)$ et $W_2(k)$ les matrices d'interconnexion dans les deux étages du réseau, il vient:

$$\Delta U(k+1)=W_2(k)W_1(k)|C-Y(k)| \qquad (2.2.2)$$

où W_1 est une matrice r×d
W_2 est une matrice diagonale r×r
$|C-Y(k)| = (|c_1-y_1(k)|, |c_2-y_2(k)|, \ldots, |c_d-y_d(k)|)^T$

W_1 peut être définit sous la forme suivante:

$$W_1 = \rho \begin{bmatrix} r_1 p_1 & r_2 p_1 & \dots & r_d p_1 \\ \dots & \dots & \dots & \dots \\ r_1 p_r & r_2 p_r & \dots & r_d p_r \end{bmatrix} = \rho \begin{bmatrix} p_1 \\ \vdots \\ p_r \end{bmatrix} [r_1 \ \dots \ r_d] \qquad (2.2.3)$$

Dans l'équation (2.2.3), ρ est un facteur multiplicatif déterminant le gain appliqué à ΔU. $r_1, r_2, \dots, r_d$ sont des réels, appelés valeurs de renforcement associées aux composantes du vecteur de sortie Y. $P=(p_1, p_2, \dots, p_r)^T \in \Re^r$, est appelé vecteur de prédiction de direction (sans sens) de ΔU avec $\sum_{i=1}^{r} p_i^2 = 1$. Le schéma de renouvellement des éléments de W_1 est énoncé en détail dans la Section VI.3.

Les éléments de W_2 déterminent le sens du vecteur ΔU à la période suivante.

$$W_2 = \begin{bmatrix} w_1 & & 0 & 0 \\ & w_2 & & \\ & & \dots\dots & \\ 0 & 0 & & w_r \end{bmatrix} \qquad (2.2.4)$$

On a $w_1 = w_2 = \dots = w_r = w$ où $w \in \{-1, 1\}$, $W_2 = wI_{r \times r}$.

Selon (2.2.2), (2.2.3) et (2.2.4), on obtient une nouvelle expression de $\Delta U(k+1)$:

$$\Delta U(k+1) = w\rho \left[\sum_{i=1}^{d} r_i \, | c_i - y_i(k) | \right] \cdot P \qquad (2.2.5)$$

La quantité de commande $\rho \sum_{i=1}^{d} (r_i \, | c_i - y_i(k) |)$ est une fonction de rétro-action qui combine linéairement les écarts $|y_i - c_i|$ à l'aide des coefficients de renforcement r_i. Dans ce sens, les r_i sont utilisés soit pour augmenter la quantité de commande lorsque l'évolution est favorable ($r_i > 0$, renforcement de la commande), soit pour diminuer la commande lorsque l'évolution de la sortie est défavorable ($r_i < 0$, pénalisation de la commande). La stratégie de commande vise à diminuer les écarts $|y_i - c_i|$ par modifications successives des r_i et application de la quantité (2.2.5). L'algorithme de modification des r_i sera présenté dans la Section VI.3.1.

Le vecteur P détermine la direction (sans sens) du vecteur ΔU. Par une méthode de discrêtisation présentée dans la Section VI.3.2, on peut trouver des P provoquant des variations angulaires quasi-orthogonales de la trajectoire de sortie. Dans ce sens, la sortie Y converge vers la consigne.

L'élément w, appelé facteur de sens du vecteur de commande, règle le sens du vecteur ΔU. Il est déterminé par l'évaluation de l'état actuel du système et des états précédents. w sera discuté dans la Section VI.4.

VI.3 ALGORITHME DU PREMIER ETAGE DU RESEAU

VI.3.1 MODIFICATION DES ri

En notant $E_i=(c_i-y_i(k-1))(c_i-y_i(k))$, $\forall i\in\{1, \ldots, d\}$, on envisage trois cas illustrés fig-2.19.

Cas 1 (fig-2.19(a)): $|c_i-y_i(k)|<|c_i-y_i(k-1)|$ et $E_i>0$.

Ce cas correspond à une évolution favorable par laquelle y_i se rapproche de c_i. Ici on envisage donc un schéma d'évolution de r_i tendant à augmenter la commande (renforcement). Ceci se traduit par les équations suivantes:

$$\begin{cases} r_i(k)=\delta r_i(k-1)+\gamma_1|c_i-y_i(k)| & \text{si } r_i(k-1)\geq 0 \\ r_i(k)=\gamma_1|c_i-y_i(k)| & \text{si } r_i(k-1)<0 \end{cases} \qquad (2.2.5)$$

dans lesquelles le coefficient $\delta\in[0,1[$ permet de conserver la trace des renforcements précédents (effet de mémoire) et le coefficient $\gamma_1>0$ permet de régler la quantité de "récompense" attribuée à la commande.

Cas 2 (fig-2.19(b)): $|c_i-y_i(k)|>|c_i-y_i(k-1)|$ et $E_i>0$.

Ce cas correspond à une évolution défavorable par laquelle y_i s'éloigne de c_i. Ici on envisage un schéma d'évolution de r_i tendant à diminuer la commande (pénalisation). Ceci se traduit par les équations suivantes:

$$\begin{cases} r_i(k)=-\gamma_2|c_i-y_i(k)| & \text{si } r_i(k-1)>0 \\ r_i(k)=\delta r_i(k-1)-\gamma_2|c_i-y_i(k)| & \text{si } r_i(k-1)<0 \end{cases} \qquad (2.2.6)$$

dans lesquelles on retrouve le coefficient δ et apparaît le coefficient $\gamma_2>0$ permettant de regler la quantité de "punition" attribuée à la commande.

Cas 3 (fig-2.19(c)): $E_i<0$

Dans ce cas, y_i traverse la valeur de consigne c_i. Ceci signifie que la quantité de commande doit décroitre. Il faut donc pénaliser en rendant $r_i(k)$ négatif. L'équation de modifica-

tion de r_i adoptée est alors la suivante:

$$r_i(k) = -\gamma_3 \frac{|c_i - y_i(k)|}{|c_i - y_i(k-1)|} \qquad (2.2.7)$$

Dans cette équation, le rapport $|c_i - y_i(k)|/|c_i - y_i(k-1)|$ permet de modifier la punition selon la règle suivante:

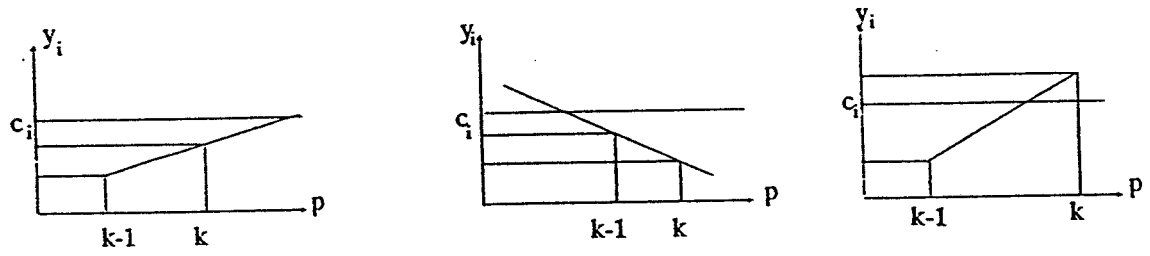
Si le dépassement de consigne s'accompagne d'un rapprochement de la consigne, la punition est moins forte que si le dépassement s'accompagne d'un éloignement de la consigne.

Le coefficient $\gamma_3 > 0$ permet de régler la quantité de punition.

γ_1, γ_2 et γ_3 affectent la convergence de l'algorithme.

Remarque: Les trois cas présentés ci-dessus conduisent à trois interprétations bien connues des automaticiens.

- 1er cas: convergence vers la consigne
- 2ème cas: divergence
- 3ème cas: dépassement de la consigne



(a) $r_i(k) > 0$ (b) $r_i(k) < 0$ (c) $r_i(k) < 0$

fig-2.19 renouvellemnt de r_i (p: numéro de période)

VI.3.2 PRINCIPE DE MODIFICATION DE P

$P=(p_1, p_2, \ldots, p_r)$ détermine la direction (sans sens) du vecteur ΔU permettant une évolution favorable de la sortie. Les critères de choix pour P sont basés sur des essais d'action-effets. A chaque période, le vecteur P le plus favorable est élu parmi un ensemble de candidats.

Pour générer les candidats, on transforme le problème de la

résolution de $(p_1, p_2, \ldots, p_r)$ avec $\sum_{i=1}^{r} p_i^2 = 1$ en celui de la résolution de $(\theta_1, \theta_2, \ldots, \theta_{r-1})$ avec $-\frac{\pi}{2} \leqslant \theta_i \leqslant \frac{\pi}{2}$, $\forall i \in \{1, \ldots, r-1\}$. La transformation de coordonnées s'exprime sous la forme suivante:

$$\begin{cases} p_1 = \sin\theta_1 \\ p_2 = \cos\theta_1 \sin\theta_2 \\ \quad \ldots\ldots \\ p_{r-1} = \cos\theta_1 \cos\theta_2 \ \ldots \sin\theta_{r-1} \\ p_r = \cos\theta_1 \cos\theta_2 \ldots \cos\theta_{r-1} \end{cases} \qquad (2.2.8)$$

La transformation ci-dessus convertit l'espace hypersphérique des $\{p_i\}$ de dimension r en l'espace hypercubique des $\{\theta_i\}$ de dimension r-1.

La procédure de génération des candidats est alors décomposée selon les étapes suivantes (cf figure 2.20 pour une illustration dans le cas r=3).

- Etape 1: L'hyper-cube des $\{\theta_i\}$ est divisé en hypercubes élémentaires résultant de la discrêtisation des variables θ_i. Chaque hypercube élémentaire est appelé "boîte" (fig-2.20(a)).

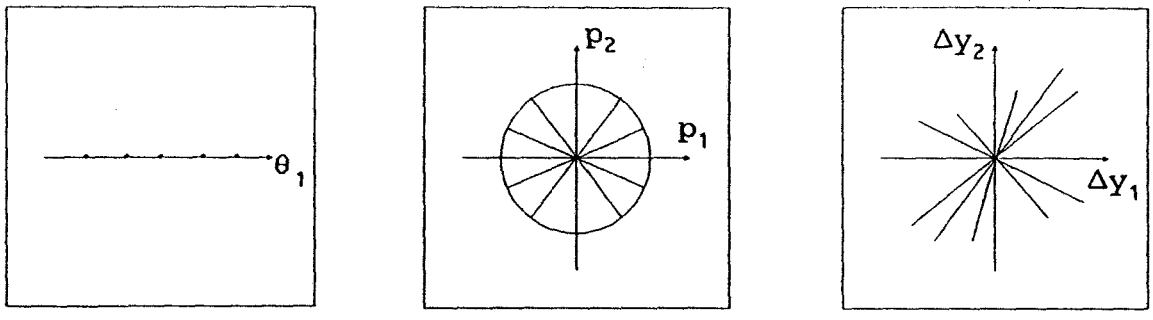
- Etape 2: A chaque boîte on affecte un vecteur de dimension r-1. Chaque vecteur a pour origine l'origine de l'espace et pour extrémité le centre de la boîte. Ces vecteurs se distribuent de manière homogène dans l'espace des $\{\theta_i\}$. On les appelle vecteurs caractéristiques.

- Etape 3: Selon l'équation (2.2.8), à chaque vecteur défini ci-dessus correspond un vecteur P. L'ensemble des vecteurs P ainsi définis recouvre l'ensemble des directions des ΔU que l'on souhaite explorer (fig-2.20(b)).

- Etape 4: Après l'application d'un ΔU au système, on obtient à la sortie un vecteur ΔY. Nous pouvons considérer que les vecteurs caractéristiques dans l'espace des $\{\theta_i\}$ permettent de recouvrir l'ensemble des directions des ΔY, y compris la direction la plus proche de la consigne C à partir de la position courante de sortie Y (fig-2.20(c)).

Plus la division de l'espace des $\{\theta_i\}$ est fine, plus les directions de ΔY sont nombreuses et mieux se fera le choix de la direction favorable. Toutefois, dans ce cas, le temps de

calcul s'allonge. Il faut donc trouver un compromis entre la division de l'espace $\{\theta_i\}$ et le temps de calcul.



(a) vecteurs caractéristiques dans l'espace $\{\theta_i\}$

(b) distribution homogène de directions dans l'espace P

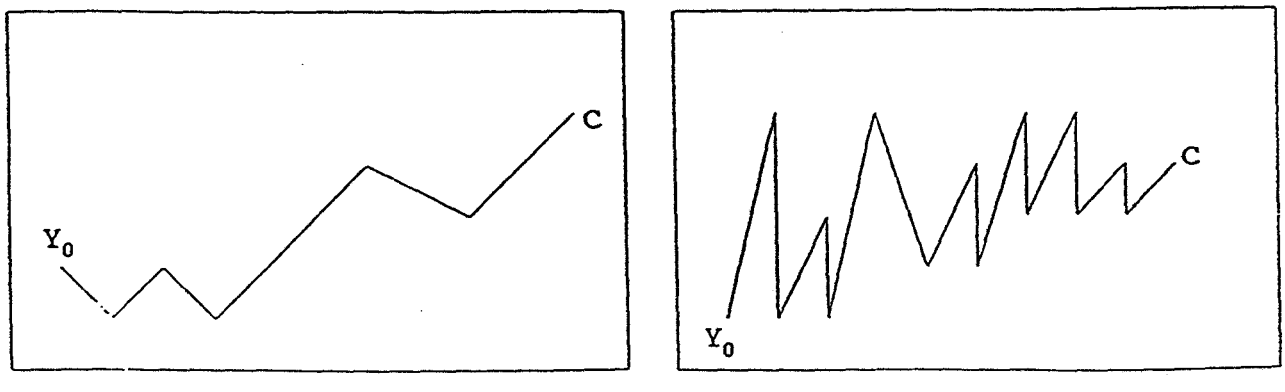
(c) distribution de directions sans sens de ΔY

fig-2.20 recherche de l'ensemble des directions P

VI.3.3 STRATEGIE DE SELECTION PARMI LES CANDIDATS

Les P sont sélectionnés de manière à provoquer des variations quasi-orthogonales sur la trajectoire de sortie. Ceci est illustré fig-2.21(a). Cette stratégie permet d'assurer une diminution rapide de l'écart entre la sortie et la consigne.

En effet, si on choisit des P qui provoquent des variations angulaires faibles sur la trajectoire de sortie (fig-2.21(b)), on provoque des oscillations qui ne permettent pas de converger rapidement vers la consigne.



(a) convergence efficace (b) convergence inefficace

fig-2.21 trajectoires de la sortie du système

La procédure de génération de P est donnée ci-dessous.

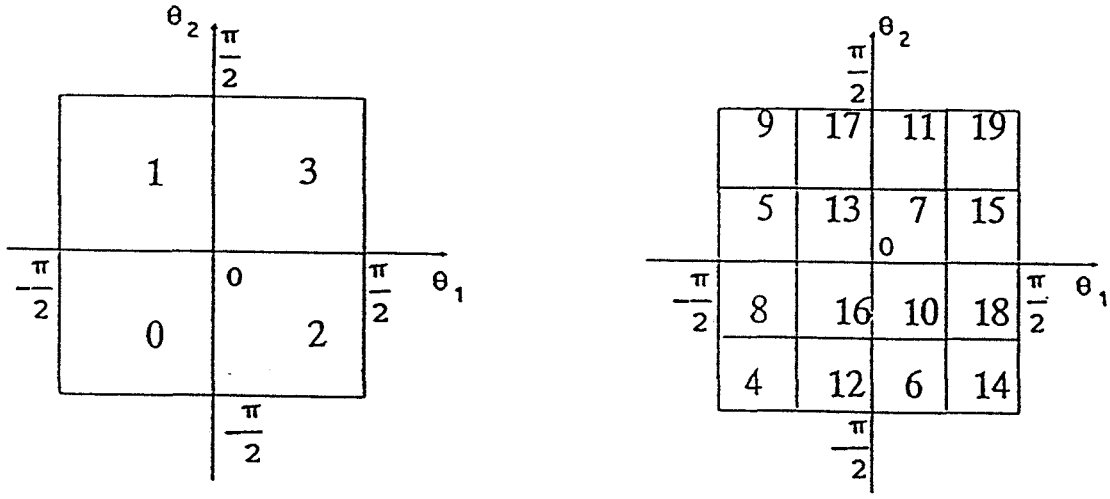
L'espace des $\{\theta_i\}$ est, dans un premier temps, divisé selon 2 niveaux. (cf fig-2.22 dans le cas r=3).

NIVEAU 1: l'espace des $\{\theta_i\}$ avec $-\frac{\pi}{2} \leq \theta_i \leq \frac{\pi}{2}$ est partitionné en 2^{r-1} boîtes disjointes dont les volumes sont identiques (fig-2.22(a)). Nous prenons, alors les centres de ces boîtes comme vecteurs caractéristiques. Les boîtes sont numérotées par $m=0,1, \ldots, 2^{r-1}-1$ et les coordonnées du vecteur caractéristique de la boîte m sont définies par $(\theta_1\ \theta_2\ \ldots\ \theta_{r-1})$ selon les règles suivantes:

m est exprimé en une combinaison binaires $\{m_i\}$ avec

$$m=m_1 2^{r-2}+ \ldots +m_{r-1} \quad \text{où } m_i \in \{0,1\},\ \forall i \in \{1,\ldots,r-1\} \qquad (2.2.9)$$

on définit les θ_i par $\theta_i = -\frac{\pi}{4} + \frac{\pi}{2} m_i \quad \forall i \in \{1,\ldots,r-1\}$.



(a) partition de niveau 1 (b) partition de niveau 2

fig-2.22 partition de l'espace des $\{\theta_i\}$ (r=3)

NIVEAU 2: Si le premier niveau ne permet pas de générer des variations orthogonales sur la trajectoire de sortie, on poursuit la subdivision de l'espace des $\{\theta_i\}$ à un deuxième niveau, selon une procédure dichotomique.

Chaque boîte du premier niveau est partitionnée en 2^{r-1} boîtes disjointes (fig-2.22(b)). On obtient alors 4^{r-1} boîtes au total. La numérotation de ces boîtes est poursuivie de $m=2^{r-1}$ à $m=2^{r-1}+4^{r-1}-1$. Le vecteur caractéristique d'une boîte quelconque m des deux niveaux est alors exprimé de la façon

suivante:

$$(\theta_1\ \theta_2\ \ldots\ \theta_{r-1}) \qquad (2.2.10)$$

avec $\theta_i = \frac{\pi}{2}m_i + \frac{\pi}{4}m_i' - \frac{\pi}{4} - \frac{\pi}{8}m_r$ pour $\forall i \in \{1, \ldots, r-1\}$

les m_i et m_i' sont des nombres binaires obtenus après décomposition de m selon les règles ci-dessous:

$$m = 2^{r-1}m' + m'' \qquad \begin{cases} m' = m_r + m_1' 2^{r-2} + \ldots + m_{r-1}' \\ m'' = m_1 2^{r-2} + \ldots + m_{r-1} \end{cases}$$

Si $m \geq 2^{r-1}$, $m_r = 1$; sinon $m_r = 0$. Et m_i, $m_i' \in \{0,1\}$ pour $\forall i \in \{1,\ldots,r-1\}$.

Dans la fig-2.22, r=3. Les boîtes du 1er niveau sont 0,1,2,3 et les vecteurs caractéristiques $(-\frac{\pi}{4},-\frac{\pi}{4})$, $(-\frac{\pi}{4},\frac{\pi}{4})$, $(\frac{\pi}{4},-\frac{\pi}{4})$, $(\frac{\pi}{4},\frac{\pi}{4})$. Les boîtes 4,5,...,19 appartiennent au 2ème niveau et les vecteurs caractéristiques sont $(-\frac{3\pi}{8},-\frac{3\pi}{8})$, ..., $(\frac{3\pi}{8},\frac{3\pi}{8})$.

Evidemment, m est une boîte du 1er niveau lorsque $m < 2^{r-1}$ et se trouve au 2ème niveau lorsque $m \geq 2^{r-1}$. A chaque période, on provoque un ΔY à la sortie par application d'une boîte m de la façon suivante:

$$m \rightarrow (\theta_1\ \theta_2\ \ldots\ \theta_{r-1}) \rightarrow P \rightarrow \Delta U \rightarrow \text{appliquer } \Delta U \text{ au système} \rightarrow \Delta Y$$

VI.3.4 ALGORITHME DE GENERATION DES P

L'algorithme de génération des P utilise une structure de données particulière comprenant:

- une file d'attente à $2^{r-1} + 4^{r-1}$ positions numérotées de 1 à $2^{r-1} + 4^{r-1}$. Ces positions contiennent les numéros des boîtes de 1er et 2ème niveaux.

- une liste à 2^{r-1} position numérotée de 1 à 2^{r-1}. Initialement, cette liste est vide.

L'algorithme a pour objectif d'affecter à la liste un ensemble de 2^{r-1} numéros de boîtes extraits de la file d'attente. Ces 2^{r-1} boîtes doivent provoquer sur la sortie des variations angulaires successives quasi-orthogonales. On sélectionne parmi ces candidats la boîte la plus favorable pour l'appliquer à la

période suivante. La procédure de génération des P est alors la suivante:

- prendre le 1er élément de la file d'attente (les autres éléments sont décalés d'une position) et le mettre dans la 1ère position de la liste et à la fin de la file d'attente. Si cet élément correspond à la boîte m, on définit b_1 :=m. Appliquer b_1 pour provoquer à la sortie un ΔY normalisé, noté V_1. Le premier P est donc calculé à partir de b_1 (cf fig-2.23).

- remplir la liste par les éléments de la file d'attente de la manière suivante:

Supposons qu'il existe j-1 éléments dans la liste($2 \leqslant j < 2^{r-1}$). Les boîtes correspondantes sont notées par b_1, b_2, ..., b_{j-1}. Les ΔY normalisés, provoqués par ces boîtes, sont notés par V_1, V_2, ..., V_{j-1}.

On sélectionne alors le j^{ème} élément de la file d'attente de la manière suivante:

A: on prend le 1er élément de la file d'attente (les autres éléments sont décalés) et on le met à la fin de la file. Si sa boîte correspondante est m, on définit b_j :=m.

B: on calcule P à partir de b_j et l'applique au système pour provoquer un ΔY, noté par V_j.

C: on teste s'il existe un V_i, provoqué par la boîte b_i de la liste ($i \in \{1,\ldots,j-1\}$), tel que $|V_i \cdot V_j| > q_j$ (c'est à dire: l'angle entre V_i et V_j est trop petit). Si V_i existe, la boîte m est rejetée et on retourne en A. Sinon, mettre b_j à la j^{ème} position de la liste.

D: j:=j+1 et retour en A. Cette procédure se répète jusqu'à $j=2^{r-1}$.

Le seuil q_i est une fonction croissante du nombre d'applications de la boîte b_j=m, noté par I_j. Après retrait de la boîte m, I_j est remis à 0.

- après avoir rempli la liste par la procédure précédente, on sélectionne un b_j pour l'appliquer à la période suivante selon la procédure ci-dessous:

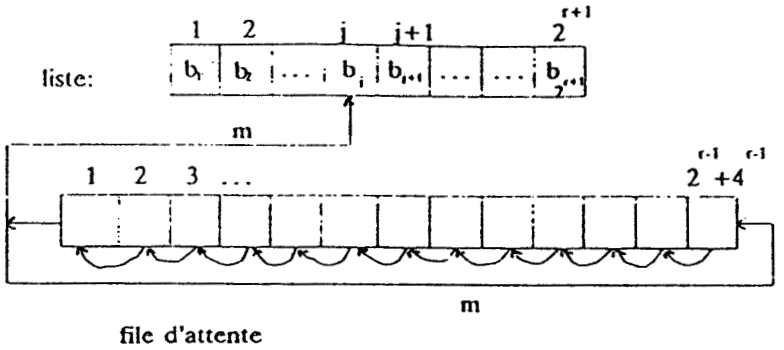


fig-2.23 sélection des boîtes à partir de la file d'attente

A': on sélectionne un b_j à partir des éléments de la liste pour que V_j, correspondant à b_j, soit le plus proche possible de la consigne à partir de la sortie actuelle Y. Le calcul s'effectue de la manière suivante:

Nous calculons respectivement les distances euclidienne de la consigne C aux vecteurs V_i passant par Y, notés par d_i (fig-2.24).

Les calculs des vecteurs dans l'espace multidimensionnel conduisent aux résultats suivants:

$$d_i=\left(\sum_{j=1}^{d}(c_j-y_j-v_{ij}\cdot f)^2\right)^{1/2} \qquad (2.2.11)$$

où $Y=(y_1\ y_2\ \ldots\ y_d)^T$, $V_i=(v_{i1}\ v_{i2}\ \ldots\ v_{id})^T$

et $f=\sum_{j=1}^{d} v_{ij}(c_j-y_j)/\sum_{j=1}^{d} v_{ij}^2$

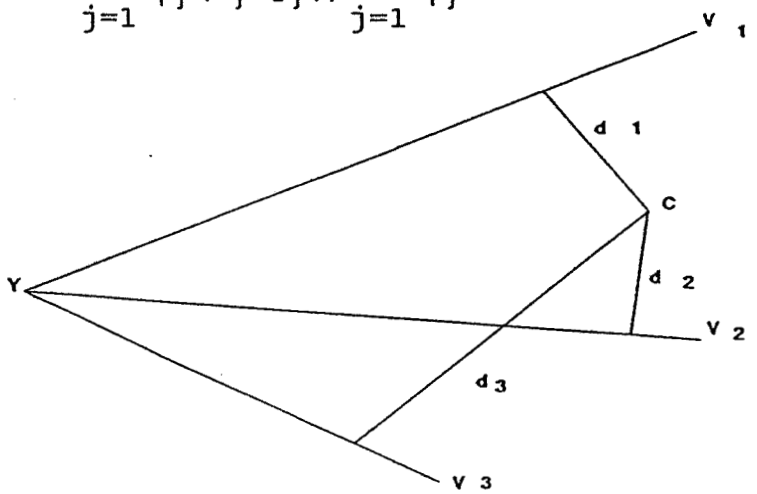


fig-2.24 calcul des distances d_i

Après l'obtention de tous les d_i ($i\in\{1, 2, \ldots, 2^{r-1}\}$), on sélectionne la boîte b_j telque $d_j=\min\{d_i\}$.

B': on effectue les étapes B et C de la procédure précédente pour tester s'il existe un V_i ($i\in\{1,\ldots,2^{r-1}\}$) tel que l'angle entre V_i et V_j sont assez petit ($|V_i \cdot V_j| > q_j$). S'il l'existe, on effectue l'étape A et retourne à B'. B' se répète jusqu'à ce que les V_i et V_j satisfont à la condition $|V_i \cdot V_j| > q_j$ ($i=1,\ldots, 2^{r-1}$), c'est à dire: les V_i sont quasi-orthogonaux à V_j.

Au départ, l'algorithme se trouve dans une phase d'apprentissage où la valeur de seuil q_j est faible et les étapes B et C sont fréquemment utilisées. Après plusieurs périodes d'essais d'action-effet, la quasi-orthogonalité des vecteurs ΔY pris deux à deux est assurée et l'apprentissage de l'environnement est réalisé pour la conduite du système par la liste ainsi définie. Dans ce cas, la valeur de q_j augmente afin que les boîtes de la liste demeurent stables.

Les prédictions $\{p_i\}$ ($i\in\{1, 2, \ldots, r\}$) sont calculées de cette manière à chaque période pour conduire le système à se diriger selon la trajectoire vers la consigne.

VI.4 ALGORITHME DU DEUXIEME ETAGE DU RESEAU

L'étage W_2 définit uniquement le sens du vecteur de commande w. w est déterminé par trois signaux d'évaluation s_1, s_2, s_3 sur le système.

En tenant compte de ces signaux d'évaluation des états du système, nous écrivons le facteur de sens de commande sous la forme suivante:

$$w=s_1 s_2 s_3 (m) \qquad (2.2.12)$$

VI.4.1 DEFINITION DE s1:

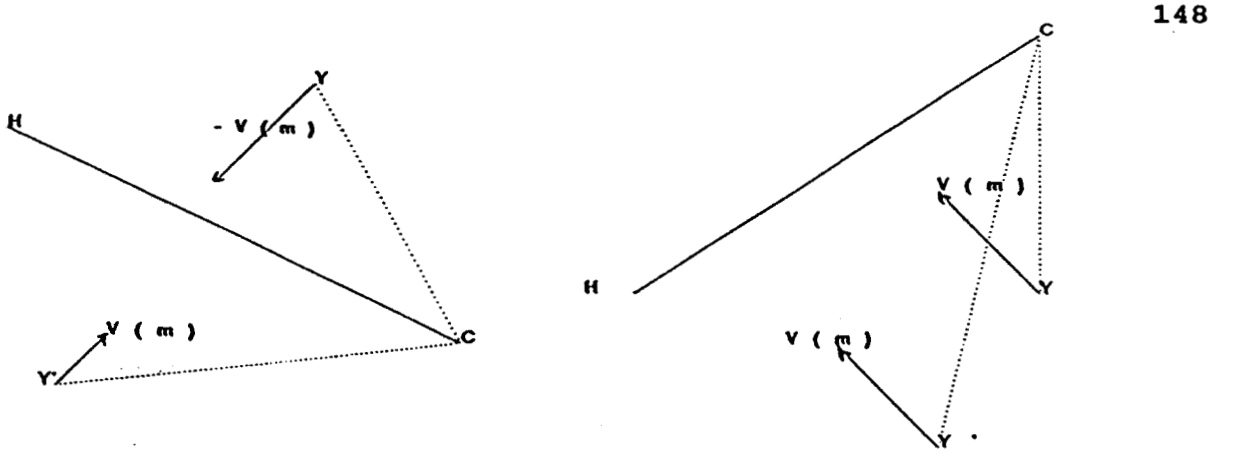
Pour éliminer l'influence du signe affecté à la quantité de commande $\sum_{i=1}^{d} r_i |c_i - y_i(k)|$, w doit contenir le signal s_1 ci-dessous:

$$s_1 = \text{sign}\left(\sum_{i=1}^{d} r_i |c_i - y_i(k)|\right) \qquad (2.2.13)$$

VI.4.2 DEFINITION DE s2:

Supposons que Y' soit la précédente sortie provoquée par le P correspondant à la boîte m dans l'espace de $\{\theta_i\}$. Son ΔY normalisé est noté par V(m).

A la période actuelle, si on souhaite une application de la boîte m à la sortie Y, les positions relatives entre Y', Y et le vecteur V(m) doivent être prises en compte dans le facteur de sens w.



(a) cotés différents (b) même coté

fig-2.25 positions relatives entre Y et Y'

Considérons le plan CH orthogonal au vecteur V(m) dans la fig-2.25. Si le produit scalaire $V(m)^T Y'C$ possède le même signe que $V(m)^T YC$, les points Y, Y' se trouvent du même coté de l'hyper-plan CH; sinon, Y, Y' se trouvent de part et d'autre de CH.

Ces deux produits scalaires peuvent se réécrire sous la forme suivante:

$$V(m)^T YC = \sum_{i=1}^{d} v_i(m)(y_i - c_i) \qquad (2.2.14)$$

$$V(m)^T Y'C = \sum_{i=1}^{d} v_i(m)(y'_i - c_i) \qquad (2.2.15)$$

Nous définissons le signal d'évaluation des positions relatives de sortie s_2 de la façon suivante:

$$s_2 = \text{sign}\left[\sum_{i=1}^{d} v_i(m)(y_i - c_i)\right] \cdot \text{sign}\left[\sum_{i=1}^{d} v_i(m)(y'_i - c_i)\right] \qquad (2.2.16)$$

Si $s_2 > 0$, Y' et Y se trouvent du même coté de l'hyper-plan CH; si $s_2 < 0$, Y' et Y se trouvent de part et d'autre de CH. s_2 sera pris en compte dans w.

VI.4.3 DEFINITION DE s3:

La commande du système s'effectue par applications alternatives des P, calculés à partir de boîtes différentes. En réalité, les effets de commande des boîtes sont différents à des

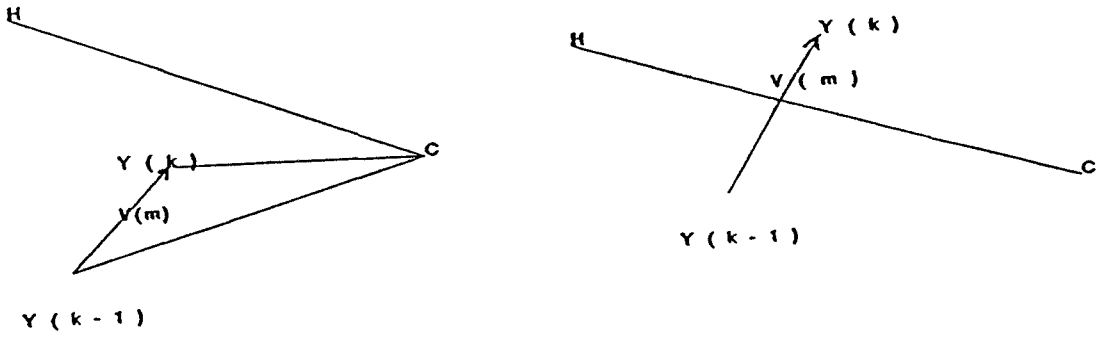
périodes différentes. Les évaluations de ces boîtes doivent être prises en compte dans la commande.

On définit la fonction d'évaluation de la boîte m par $s_3(m)$, calculée ci-dessous:

Initialement, $s_3(m)=1$. En général, si la boîte m a été appliquée à la période k, $s_3(m)$ sera renouvelée, après cette application, sous la forme suivante:

$$\begin{cases} s_3(m)=1 \quad \text{si} \quad \text{sign}\left[\sum_{i=1}^{d} v_i(m)(c_i-y_i(k-1))\cdot\sum_{i=1}^{d} v_i(m)(c_i-y_i(k))\right] \\ \qquad\qquad +\text{sign}(\text{norm}(k-1)-\text{norm}(k))=2 \\ s_3(m)=-1 \quad \text{sinon} \end{cases}$$

où $\text{norm}(k)=\|C-Y(k)\|$ (2.2.17)



(a) même coté (b) cotés différents

fig-2.26 positions relatives entre Y(k) et Y(k-1)

Evidemment, $s_3(m)$ est le signal d'évaluation des positions relatives entre Y(k-1) et Y(k). $S_3(m)=1$ signifie que le déplacement de Y(k-1) à Y(k) est favorable et le sens de V(m) doit être maintenu dans l'application suivante de m. $s_3(m)=-1$ signifie que ce déplacement est défavorable ou que l'hyper-plan CH est traversé à la période k. Le sens de V(m) doit être inversé.

VI.5 RESULTATS DE SIMULATION ET ANALYSE DE CONVERGENCE

VI.5.1 PRINCIPLE DE SIMULATION

Comme l'algorithme de l'automate stochastique, on introduit deux fenêtres d'observation des échantillons à chaque période. Chaque fenêtre contient n échantillons de la sortie du système. Les données dans la $2^{ème}$ fenêtre peuvent être utilisées pour estimer le vecteur de sortie stationnaire M . Nous choisissons la moyenne des échantillons de cette fenêtre comme la sortie à la période courante du système (cf la Section V.5).

Les fonctions d'évaluation concernant les 2 étages du réseau ci-dessus sont renouvelées à partir des vecteurs Y, ΔY et des données précédentes.

La procédure de simulation de cet algorithme se fait de la façon suivante:

Etape 1: Initialisation de W_1 et W_2 en assignant w=1, $r_i=0$ pour $\forall i \in \{1,\ldots,d\}$. $\{p_i\}$ initial correspond à la boîte 0 dans l'espace $\{\theta_i\}$.

Etape 2: Application de ΔU au système et obtention d'une nouvelle position Y et d'une nouvelle direction ΔY après filtrage des données par fenêtrage.

Etape 3: Test de la condition $\|Y-C\|<\varepsilon$ (ε est l'écart désiré). Si cette condition est satisfaite, arrêter; sinon, aller à 4.

Etape 4: Renouvellement de W_1 et W_2 par les schémas définis dans les sections précédentes. Retour à 2.

Nous avons réalisé la simulation sur des systèmes linéaires multi-variables. Les dimensions des sorties et des entrées des systèmes sont respectivement 2, 3, 4. La simulation a été effectuée sur un IBM/PC-AT (f=12Mz) en langage PASCAL. Tous les exemples montrent une bonne convergence de cet algorithme.

VI.5.2 RESULTATS DE SIMULATION

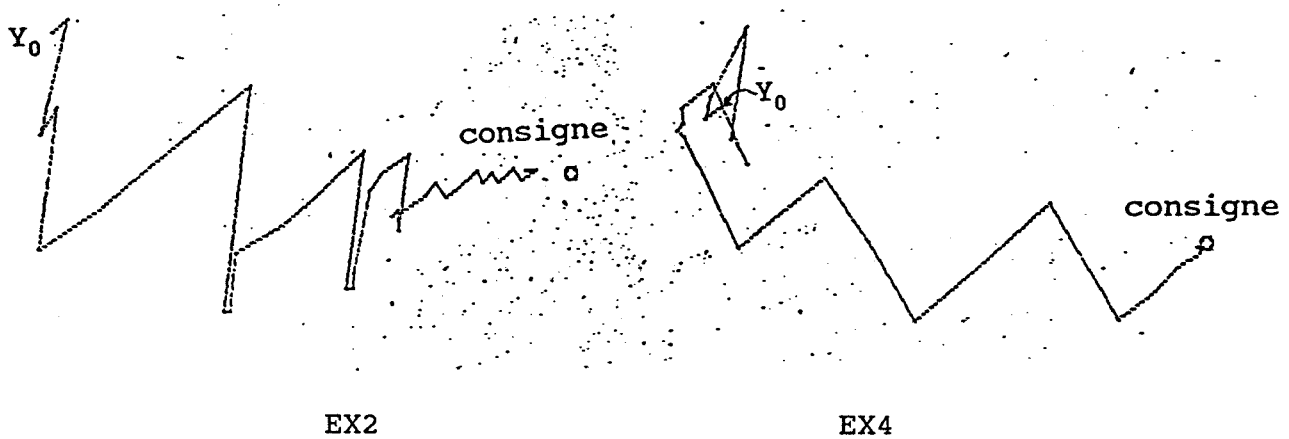
Nous montrons ci-dessous les résultats de simulation pour

les exemples du Chapitre V.

simulation	d	r	Consigne	Y(fin)	U(fin)	Y(début)	nb
EX1	1	2	20	20.05	$\begin{bmatrix}5.23\\2.15\end{bmatrix}$	-0.05	29
EX2	2	2	$\begin{bmatrix}10\\8\end{bmatrix}$	$\begin{bmatrix}9.91\\7.97\end{bmatrix}$	$\begin{bmatrix}-21.4\\34.8\end{bmatrix}$	$\begin{bmatrix}-0.03\\-0.08\end{bmatrix}$	36
EX3	3	3	$\begin{bmatrix}10\\12\\14\end{bmatrix}$	$\begin{bmatrix}9.39\\12.1\\14.1\end{bmatrix}$	$\begin{bmatrix}-19.3\\34.5\\7.35\end{bmatrix}$	$\begin{bmatrix}-0.09\\0.34\\-0.02\end{bmatrix}$	49

Tableau-2.2 valeurs numériques de simulation

On donne ci-dessous les trajectoires de convergence pour les deux exemples EX2 et EX4 avec d=2 et r=2.



EX2 EX4

fig-2.27 résultats graphiques de simulation

Evidemment, au départ de la procédure de commande, les vecteurs ΔY sont différents à chaque période, ce qui signifie que l'apprentissage des poids de prédiction de direction $\{p_i\}$ s'effectue par des essais d'actions-effets sur le système et par renouvellements consécutifs des réseaux neuronaux. Après plusieurs périodes, les directions de sortie deviennent stables et quasi-orthogonales. Les 2^{r-1} boîtes sont bien sélectionnées par apprentissage de l'environnement. Dans tous les exemples de simulation, les trajectoires de la sortie Y convergent vers les consignes désirées prédéfinies.

VI.5.3 ANALYSE DE CONVERGENCE DE L'ALGORITHME

La convergence de l'algorithme dépend des choix des para-

mètres suivants:

ρ, q_i ($\forall i \in \{0, 1, \ldots, 2^{r-1}-1\}$), γ_i ($i=1, 2, 3$), δ, etc.

ρ est un coéfficient de quantité de commande. Si ρ est trop grand, l'algorithme risque de diverger vers l'infini. Si ρ est trop petit, la variation de sortie provoquée par la commande n'est pas évidente à cause des perturbations des échantillons de la sortie. On doit donc choisir des ρ convenables pour éviter la divergence de la sortie et pour éliminer les perturbations des échantillons. Dans notre simulation, nous choisissons $\rho=0.05$.

Les seuils q_j sont également importants pour la convergence de l'algorithme au cours de commande. Dans la simulation, on définit

$$q_j = \min\{0.95,\ 0.9(0.99+0.01I_j)\} \quad \text{pour } d=2 \qquad (2.2.18)$$

$$q_j = \min\{0.99,\ 0.92(0.97+0.03I_j)\} \quad \text{pour } d=3 \qquad (2.2.19)$$

Les autres paramètres γ_i, δ affectent également la convergence de l'algorithme. Dans chaque système spécifique, il est necéssaire de déterminer ces paramètres a priori pour assurer la convergence du système.

VI.5.4 REMARQUES FINALES

L'approche que nous avons proposée est différente des études générales sur les réseaux neuronaux adaptatifs. Pendant certaines périodes de commande, l'apprentissage à travers le réseau s'effectue par des essais d'action-effets du système où les actions sont générées par des fonctions d'évaluation des états du système. Le réseau est capable de fonctionner dans des espaces multi-variables d'entrée et de sortie.

Les principes de commande par apprentissage décrits pourront être utilisés dans des domaines variés. Dans [MILLER, 1987], on établit une architecture de commande pour les systèmes robotiques complexes avec multi-boucles de retour de capteurs. Au sein de l'architecture de commande, un algorithme d'apprentissage général est utilisé pour régler les relations entre sorties de capteurs et variables de commande des systèmes. Cette méthode peut être combinée directement avec les autres techniques de commande existantes. Par exemple, on peut introduire des réseaux à apprentissage de renforcement et de supervision pour raffiner la performence des régulateurs de commande prédéfinis.

La simulation montre la bonne convergence de l'algorithme. Pourtant, il reste certains problèmes non-résolus dans la commande de système plus complexes. Cet algorithme est basé sur l'hypothèse que le changement de direction du vecteur de sortie, qui correspond à une entrée fixée, est lent et stable. Dans les systèmes non-linéaires et complexes, cet algorithme doit être modifié pour s'adapter à l'environnement spécifique.

CONCLUSION

Les systèmes de commande traditionnels dépendent généralement de l'intervention humaine pour traiter des grandes incertitudes. Pourtant, l'intervention humaine est inacceptable dans certaines applications temps réel. Il est donc nécessaire de proposer de nouvelles techniques automatiques pour effectuer le traitement des grandes incertitudes.

Les techniques de commande adaptative classiques sont développées pour les systèmes qui possèdent de grandes incertitudes dues aux variations des paramètres et de l'environnement. Dans ces techniques, deux boucles de retour sont utilisées pour réaliser la commande du système. La première boucle possède la capacité de suivre les paramètres du système, et la deuxième boucle génère la commande pour obtenir des performances acceptables du système.

Lorsque l'incertitude du système augmente, les techniques de commande adaptative présentées ci-dessus ne sont plus disponibles et on utilise souvent des approches de commande intelligentes pour conduire les systèmes [FU, 1971].

Actuellement, quatre approches sont développées pour réaliser la commande intelligente du système:

(1) Systèmes experts comme éléments adaptatifs dans la commande [ASTRÖM, 1983], [ASTRÖM, 1986], [BETTA et LINKENS, 1990], [DORAISWAMI et JIANG, 1989];

(2) Calculs flous comme éléments de génération de décision dans la commande [YING, SILER et BUCKLEY, 1990], [ANDERSON et NIELSEN, 1985];

(3) Automate stochastique comme éléments de régulation dans la commande;

(4) Réseaux neuronaux comme éléments de compensation dans la commande.

Les algorithmes de pilotage présentés dans les Chapitres V et VI, correspondant respectivement aux cas (3) et (4), sont

basées sur l'apprentissage au sein de régulateurs de commande. En effet, l'objectif de l'apprentissage est d'acquérir de nouveaux comportements ou d'en modifier d'autres par l'expérience. Les premiers travaux de l'apprentissage ont été effectués en intelligence artificiel selon deux tendances: les méthodes heuristiques et/ou symboliques, et les méthodes numériques. Les deux approches présentées dans ce mémoire se situent dans le cadre des méthodes numériques et possèdent des caractéristiques de l'IA.

Pour l'instant, la tendance numériques s'est développée dans les domaines de la reconnaissance des formes et de la commande intelligente. Automate stochastique et réseaux neuronaux sont deux techniques importantes dans cette tendance. Des lois physiques ou des propriétés mathématiques peuvent être mise en évidence par application des algorithmes d'apprentissage.

Les deux approches présentées dans ce mémoire peuvent être comparées sous divers aspects:

- La convergence de l'algorithme à réseaux neuronaux est plus rapide que celle de l'automate stochastique (cf tableau-2.1 et tableau-2.2).

-Les oscillations sont plus nombreuses dans la trajectoire de l'algorithme de l'automate stochastique.

- L'algorithme à réseaux neuronaux est en général plus performant si les paramètres sélectionnés sont appropriés.

- L'algorithme de l'automate stochastique est plus simple que celui des réseaux neuronaux et ce dernier possède plus de risques de divergence.

BIBLIOGRAPHIE

[BRO 74] William L. BROGAN
Modern Control Theory
QUANTUM PUBLISHERS, INC., 1974

[FAR 86] C.FARGEON
Commande numérique des systèmes: applications aux engins mobiles et aux robots
Paris, Masson, 1986

[NAJ 82] K.NAJIM
Commande adaptative des processus industriels
Paris, Masson, 1982

[NAR 74] KUMPATI S. NARENDRA and M.A.L. THATHACHAR
"Learning automat --- a survey"
IEEE Trans. Syst. Man. Cybern., vol SMC-4, no.4, July 1974

[THA 85] M.A.L. THATHACHAR and P.S. SASTRY
"A new approach to the design of reinforcement scheme for learning automata"
IEEE Trans. Syst. Man. Cybern., vol. SMC-15, no.1, January/February 1985

[SIM 89] RAHUL SIMHA and JAMES F. KUROSE
"Relative reward strength algorithme for learning automata"
IEEE Trans. Syst. Man. Cybern. vol.19, no.2, March/April, 1989

[NAR 77] KUMPATI S. NARENDRA, E. ALLEN WRIGHT and LORNE G. MASON
"Application of learning automata to telephone traffic routing and control"
IEEE Trans. Syst. Man. Cybern., vol. SMC-7, no.11, november, 1977

[AND 89] Charles W. ANDERSON
"Learning to Control an Inverted Pendulum Using Neural Networks"
IEEE Control Systems Magazine, pp.31-37, April 1989.

[BAR 83] ANDREW G. BARTO, RICHARD S. SUTTON and CHARLES W. ANDERSON
"Neurolike Adaptive Elements That Can Solve Difficult Learning Control Problems"
IEEE Trans. Syst., Man, Cybern., vol. SMC-13, no.5, Sept.-Oct. 1983

[NIN 89] Geoffrey E. HINTON
"Connectionist Learning Procedure"
Artificial Intelligence 40 (1989), pp.185-234.

[PSA 1988] D. PSALTIS, A. SIDERIS and A. A. YAMAMURA
"A Multilayered Neural Network Control", IEEE Control System Magazine, pp.17-21, April 1988.

[DAV 89] E. DAVALO et P. NAIM
Des réseaux de neurones
1989, EYROLLES.

[BAV 88] B. BAVARIAN
"Introduction to Neural Networks for Intelligent Control"
IEEE Control System Magazine, pp.3-7, April 1988.

[MIL 87] W. T. MILLER
"Sensor-Based Control of Robotic Manipulators Using a General Learning Algorithm"
IEEE Joural of Robotics and Automation, vol. RA-3, no.2, April 1987.

[FU 71] K.S.FU
"Learning control systems and intelligent control systems: an intersection of artificial intelligence and automatic control"
IEEE Trans. Aut. Control, AC-16(1), 70-72.

[AST 83] K.J.ASTRÖM
"Theory and application of adaptive control"
Automatica, 19, pp.471-486, 1983

[AST 86] K.J.ASTRÖM, J.J.ANTON and K.E.ARZEN
"Expert control"
Automatica, vol.22, no.3, pp.277-286, 1986

[BET 90] A.BETTA and D.A.LINKENS
"Intelligent knowledge-based system for dynamic system identification"
IEE PROCEEDINGS, vol.137, Pt.D, no.1, January, 1990

[DOR 89] R.DORAISWAMI and J.JIANG
"Performance monitoring in expert control systems"
Automatica, vol.25, no.6, pp.799-811, 1989

[AND 85] T.R.ANDERSEN and S.B.NIELSEN
"An efficient single output fuzzy control algorithm for adaptive applications"
Automatica, vol.21, no.5, pp.539-545, 1985

[YIN 90] H. YING, W. SILER and J. J.BUCKLEY
"Fuzzy control theory: a nonlinear case"
Automatica, vol.26, no.3, pp.513-520, 1990

CONCLUSION GENERALE

Ce mémoire se situe dans le cadre de l'étude des systèmes en vue de leur pilotage par observation des échantillons statistiques delivrés en sortie de ces systèmes.

L'étude comporte deux volets:

- analyse de la structure du processus (Partie I)

- pilotage de ce processus (Partie II).

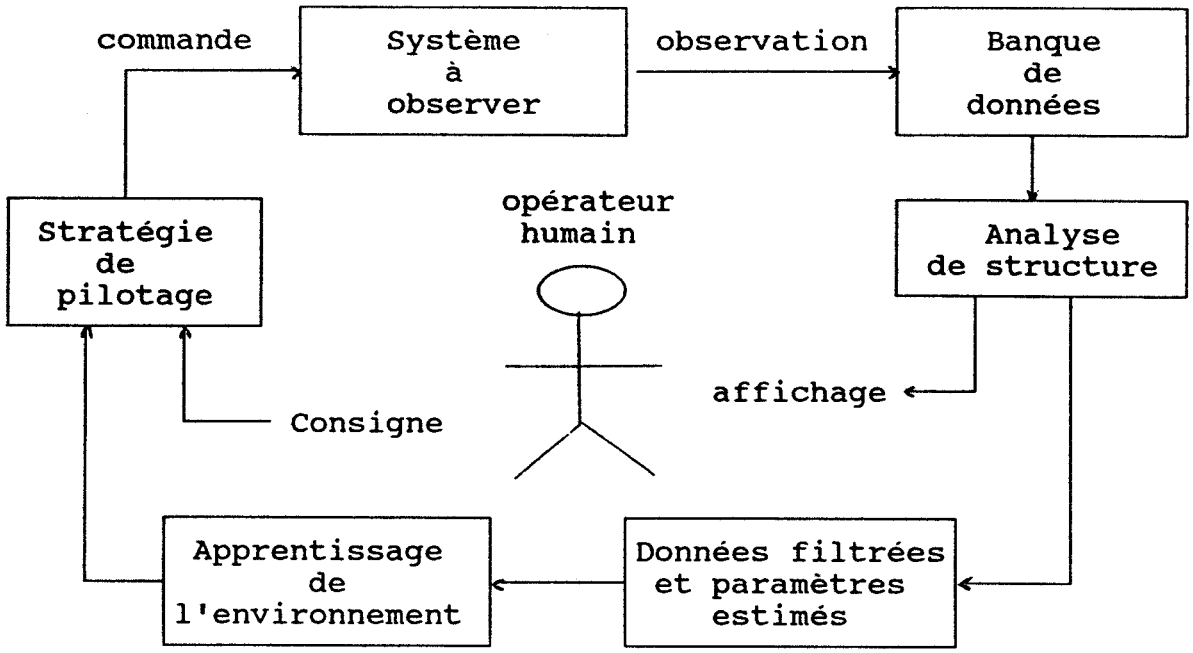
Le pilotage est fondée sur l'exploitation des caractéristiques statistiques du processus. Dans ce sens, on a montré d'abord que l'observation du processus dynamique, pouvait se ramener à la caractérisation du comportement du processus. A partir de cette observation, on a élaboré les stratégies de pilotage permettant de conduire le processus vers une structure caractérisant la consigne.

Pour réaliser cet objectif, de nombreuses méthodes ont été mises en oeuvre:.

Pour l'analyse de la structure du processus, on introduit la notion de fenêtrage dans l'estimation récursive des paramètres statistiques et dans la classification automatique par scission. On applique également la méthode du maximum de vraisemblance modifiée dans l'estimation des paramètres à partir d'échantillons fortement perturbés. Par ailleur, une procédure de fusion, redéfinissant l'indice interne selon la séparabilité et la compacité des nuages d'échantillons, permet de trouver la meilleure partition pour tous les échantillons.

Pour le pilotage du processus, on met en oeuvre respectivement un automate stochastique et un réseau neuronal pour effectuer l'apprentissage de la stratégie de pilotage. La méthode de fenêtrage est également utilisée pour filtrer les échantillons observés.

L'étude effectuée dans ce mémoire constitue alors le fondement d'un module d'auto-organisation appartenant à une boucle de régulation, illustré sur la figure ci-dessous:



Boucle d'analyse et de pilotage

Les algorithmes proposés dans ce mémoire permettent à l'utilisateur de réaliser toutes les fonctions de cette boucle. Selon le schéma, il est possible d'intervenir sur la boucle à partir de l'extérieur par modification de la consigne. Il est également possible d'observer les résultats de l'analyse de structure. Par conséquent, cette boucle peut être considérée comme un dispositif d'auto-organisation avec possibilité d'intervention d'un opérateur humain.