Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES MATHÉMATIQUES

Utilisation des tests d'hypothèses dans la méthode des hypervolumes pour la détermination du nombre de classes en classification automatique

Pierard, Anne

Award date:
1992

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 19. Aug. 2024

Facultés Universitaires Notre-Dame de la Paix
FACULTE DES SCIENCES
Rue de Bruxelles 61 - 5000 NAMUR
Tél. 081/72.41.11 - Telex 59222 facnam-b - Telefax 081/23.03.91

André HARDY
Département de Mathématique
Facultés Universitaires de Namur
Rempart de la Vierge 8
B - 5000 NAMUR - BELGIUM

Utilisation des tests d'hypothèse dans la méthode des hypervolumes pour la détermination du nombre de classes en classification automatique

Mémoire présenté pour l'obtention du grade
de Licencié en Sciences
Mathématiques
par

PIERARD Anne

Promoteur : Professeur J.-P. RASSON

Juin 1992

Je tiens à remercier Monsieur le Professeur J.P. Rasson, promoteur de ce mémoire, de m'avoir permis d'effectuer ce travail et pour les conseils donnés au cours de sa réalisation.

Mes remerciements vont également à Monsieur T. Kubushishi pour sa disponibilité, ses encouragements et l'aide apportée dans le cadre de ce mémoire.

Je suis aussi profondément reconnaissante envers ma famille, et plus particulièrement envers mes parents, qui m'ont permis d'entreprendre ces quatre années d'études, envers Philippe mon mari, pour son soutien et sa collaboration et tous ceux qui d'une manière ou d'une autre, ont participé à l'élaboration de ce mémoire.

Résumé

Nous proposons de résoudre le problème de la détermination du nombre de classes de la partition optimale en classification automatique par le test d'hypothèse, en utilisant l'algorithme basé sur le critère des hypervolumes et par la maximisation du vide. Nous comparons le test d'hypothèse et la méthode du coude sur de nombreux exemples.

Abstract

Our purpose is to resolve the problem in number of clusters determination for the optimal collection by hypothesis test. We use the algorithm based on hypervolumes criterion and we compare the hypothesis test with another method called "la méthode du coude" taking a lot of exemples.

Introduction

A moins de huit ans de l'an 2000, nous vivons dans une société où tout, ou presque, est régi par l'ordinateur. Qui dit ordinateur, dit ensemble de données, données qu'il convient de classifier pour les rendre "lisibles".

Au fil des années, les techniques de classification se sont développées et affinées. Parmi celles-ci, la classification automatique dont le but est de diviser une population donnée d'objets décrits par un ensemble de variables en un nombre relativement restreint de sous-groupes.

Grâce aux techniques utilisées, nous cherchons la partition en k classes qui minimise un critère.

Dans le premier chapitre de ce mémoire, nous rappelons ce qu'est l'algorithme basé sur la technique de maximisation du vide développée par V. Gendarme. Ensuite, nous posons le problème de la détermination du nombre de classes de la partition optimale.

Une solution possible, c'est le test d'hypothèse qui, après un rappel de la théorie du processus de Poisson, constitue l'essentiel du chapitre 2.

Un test d'hypothèse pour lequel il convient d'envisager les difficultés pratiques mais aussi certaines remarques théoriques. C'est l'objet de notre chapitre 3.

Un cheminement qui nous conduit tout naturellement à aborder un certain nombre d'applications des plus simples aux plus complexes, des données de référence aussi. Toutes ces applications sont développées dans le chapitre 4.

Enfin, le chapitre 5 aborde un nouvel algorithme qui doit nous permettre de résoudre les problèmes pour lesquels il subsiste encore des difficultés.

Chapitre 1

Classification automatique

1.1 Introduction

La nature offre un grand nombre d'objets qu'il est souhaitable de répartir en catégories. Nombre de disciplines scientifiques sollicitent des classifications. En médecine, il s'agit par exemple, de découvrir les principaux regroupements de malades ayant le même comportement vis-à-vis de certains paramètres retenus pour les caractériser. En botanique, il s'agit de mettre par exemple, en évidence des sous-espèces d'une même variété à partir d'un tableau de données où des plantes sont caractérisées par un certain nombre de mesures.

L'objectif est de regrouper ensemble les données ou variables dans des classes telles que les éléments à l'intérieur d'une classe soient naturellement associés entre eux et cependant relativement distincts de ceux d'une autre classe. L'approche du problème et les résultats qui en découlent dépendent principalement du fait de savoir comment l'investigateur cherche à rendre opérationnel le sens des expressions "association naturelle" et "relativement distinct".

L'information apportée par une classification doit atteindre un résultat profitable. Pourquoi ? Pour de multiples raisons. Des regroupements inattendus apparaissent, permettant de faire de nouvelles hypothèses. En revanche, des regroupements attendus n'existent pas, ce qui fait ressortir le faible pouvoir séparateur des paramètres utilisés. Les classes obtenues et leurs imbrications assurent une vue concise et structurée des données. Les classes significatives entraînent la définition de fonctions de décision permettant d'attribuer un nouvel objet à la classe dont il est proche, etc.

1.2 Le problème de classification

1.2.1 Formalisation

Le problème de la classification peut souvent s'énoncer en termes d'optimisation :
soit une fonction W définie sur un ensemble de partitions et à valeurs réelles. Il s'agit de trouver la partition P^* minimisant ou maximisant cette fonction. La complexité du problème naît de son caractère combinatoire. Le nombre de solutions est astronomique dès que le nombre de points à classer dépasse 20. Pour fixer les idées, rappelons quelques chiffres significatifs. Le nombre de partitions de n objets en k classes non vides S(n,k) est égal à :

$$S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^{k-i}\frac{k!}{i!(k-i)!}i^n$$

et est appelé nombre de *Stirling* du deuxième ordre.

Nous avons par exemple $S(100,5) \simeq 10^{68}$ ou $S(60,2) \simeq 10^{18}$

Aussi, la nécéssité de trouver des méthodes capables de fournir automatiquement une bonne classification sans devoir évaluer toutes les solutions possibles apparaît dès qu'il s'agit de classifier plus d'une dizaine d'objets.
Formellement,

soit$E = \{x_1, \ldots, x_n\}$, l'ensemble des individus à classer.
On suppose que l'ensemble E est un espace vectoriel euclidien fini et que $x_i \in I\!R^{DIM} \forall$ i.

Le problème de classification devient :

Trouver une partition de l'ensemble E en k classes disjointes $C_1, C_2, \ldots, C_k$ avec k fixé.
Soit P_k l'ensemble de toutes ces partitions en k classes.
A tout P dans P_k, nous associons la valeur d'un critère de classification W qui mesure la qualité de chaque partition P.
Le problème est alors de trouver la partition qui minimise la valeur du critère parmi l'ensemble des partitions en k classes.

Dans le cas du critère des hypervolumes, aussi appelé "critère de Rasson" , on suppose que l'ensemble E des individus à classer est une réalisation d'un processus de Poisson homogène dans l'union C de k domaines convexes compacts disjoints $C_1, C_2, \ldots, C_k$.

Le critère de classification W considéré est le suivant : la somme des mesures de Lebesgue des enveloppes convexes des classes.

Le problème s'écrit alors :

$$W : P^k \longrightarrow \mathbb{R}^+ : P = \{C_1, \ldots, C_k\} \rightsquigarrow W(P) = \sum_{i=1}^{k} m(C_i)$$

où $m(C_i)$ représente la mesure de Lebesgue de l'enveloppe convexe de la classe C_i.

On cherchera alors la partition P^* telle que

$$W(P^*) = \min_{P \in P_k} \sum_{i=1}^{k} m(C_i)$$

Du fait de l'additivité du critère choisi [2], toute sous-partition d'une partition optimale est optimale sur son ensemble de définition. Ce qui revient à dire que si on supprime de l'échantillon les individus d'une des k classes de la partition, alors la classification appliquée aux individus restants fournit les (k-1) autres classes.

Remarquons enfin que les enveloppes convexes compactes des classes obtenues sont disjointes.

1.2.2 Méthode par maximisation du vide

Rappelons brièvement en quoi consistait l'algorithme de V. Gendarme [5]. Il nous servira de base de travail pour la suite.

Celui-ci est basé sur la technique de maximisation du vide et sur le critère des hypervolumes.

Il comporte deux étapes :

Première étape : division.

- Division en deux :

  On construit, autour de toutes les données, un hyperrectangle suivant la base canonique de l'espace euclidien $\mathbb{R}^{DIM}$. Chaque face de l'hyperrectangle est supportée par un hyperplan. Pour chaque paire de faces non contigües de l'hyperrectangle, on cherche deux hyperplans séparateurs. Ceux-ci seront parallèles aux faces considérées et permettront d'isoler un hyperrectangle de vide dans l'ensemble de points. Lors de la division, on veillera à supprimer le plus grand hyperrectangle de vide. Ceci déterminera les deux classes.

- Division successive en v classes :

  On divise alors la classe pour laquelle le partitionnement engendre la plus grande augmentation du volume de vide et cela jusqu' à l'obtention de v classes (v fixé).

Deuxième étape : recollement.

Cette étape permet de rassembler les classes pour lesquelles le critère de recollement est minimal.

Ce critère de recollement correspond à la mesure de Lebesgue de l'écart entre les deux classes considérées. Autrement dit, pour chaque paire de classes C_i et C_j, on calcule :

$$m(écart_{ij}) = m(C_i \cup C_j) - (m(C_i) + m(C_j))$$

où m(.) représente la mesure de Lebesgue de l'enveloppe convexe de (.).

On recolle successivement les deux classes pour lesquelles le critère de recollement est minimal, et ce jusqu' à l'obtention de k classes.

Remarquons, qu' ici, nous abandonnons la notion d'hyperrectangle et considérons la mesure de l'enveloppe convexe des classes obtenues. Le calcul de l'enveloppe convexe est effectué par la méthode exacte décrite dans le mémoire de L. Waerenburgh [12] .

Dans la suite de ce mémoire, nous conserverons les notations v et k respectivement pour le nombre de classes obtenues après divisions et le nombre de classes obtenues après recollement.

Toutefois, la méthode nécessite le choix du nombre de classes k. Il est demandé à l'utilisateur. Mais lorsque l'on traite un ensemble de données, on ne sait pas a priori le nombre de classes qui composent la partition optimale.

La méthode utilisée dans son mémoire est appelée "méthode du coude". Elle consiste à examiner la variation du critère de classification W en fonction du nombre de classes.

La présence d'un "coude" dans la courbe formée par W indique une stabilisation de la valeur du critère optimisé. Le nombre de classes à retenir sera dès lors indiqué par le coude le plus marqué de cette courbe.

Ici, le critère W utilisé est celui des hypervolumes :

$$W(P) = \sum_{i=1}^{k} m(C_i)$$

La méthode du coude indique que la partition en trois est la meilleure.(FIG. 1.1)

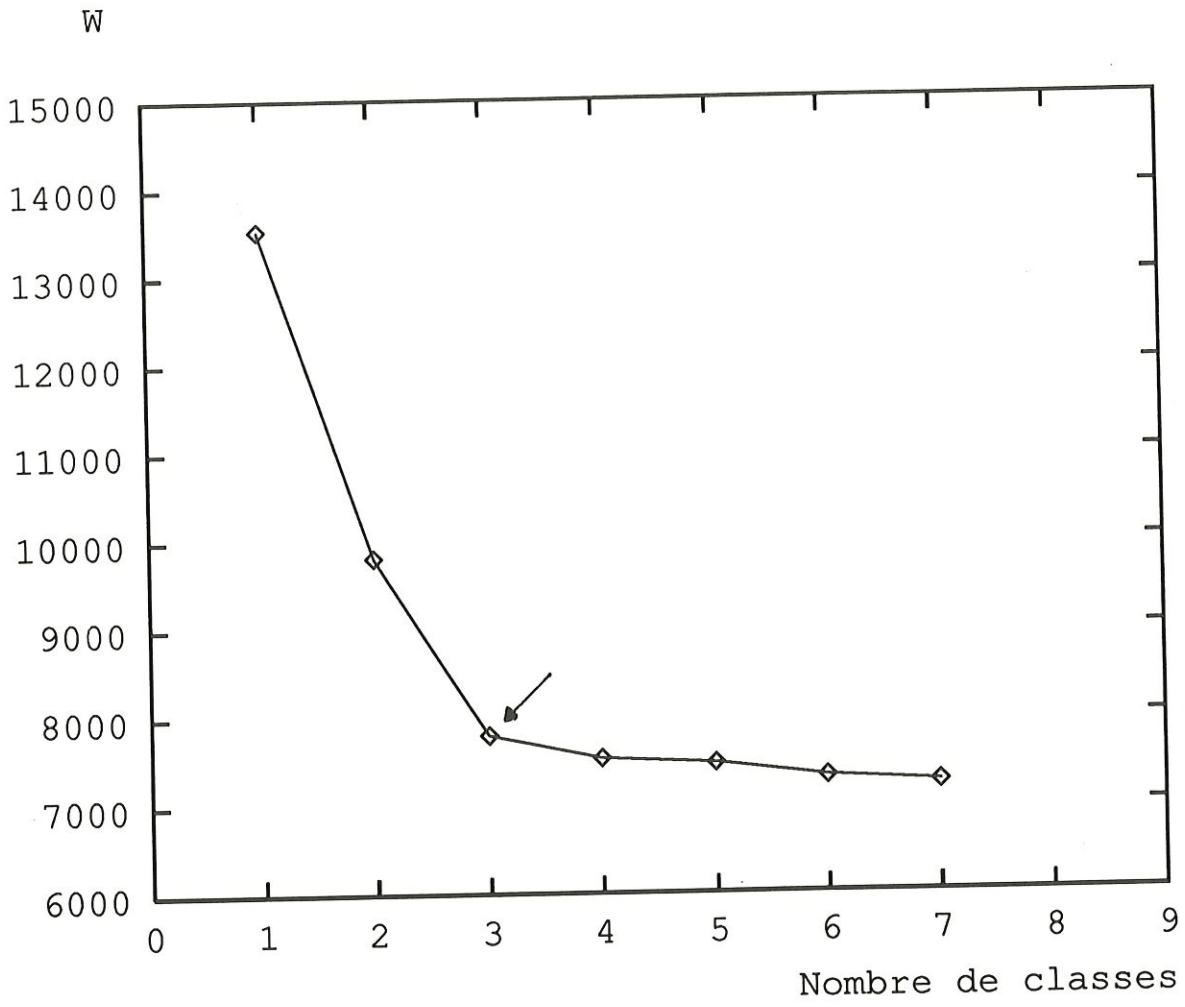


*FIG.*1.1

Dans de nombreux cas, la méthode s'avère pertinente pour déceler le nombre optimal de classes. Elle n'est cependant pas infaillible et ne parvient pas à satisfaire tous les chercheurs.

Théoriquement, il faudrait conclure à l'absence de structure lorsque le graphe décroît de façon constante et monotone. Et il devrait toujours y avoir une structure lorsque nous observons un "coude". La réalité est tout autre. Un graphe monotone correspond parfois à une structure dans les données . D'autant plus que l'observation d'un coude sur un graphe apparaît souvent subjective et peut-être faussée par le choix du repère.

La recherche du nombre de classes de la partition optimale n'est donc pas chose aisée. Nous allons le voir:

Voici les données de Ruspini (FIG. 1.2). Elles sont souvent utilisées en classification automatique pour tester de nouvelles méthodes.

*FIG.*1.2

A première vue, le nombre de classes de la partition optimale n'est vraiment pas évident à déterminer. Nous hésitons entre 3 ou 4 classes. Nous pourrions même envisager 5 classes.

Pour résoudre ce problème, la méthode du coude (FIG. 1.3) n'est certainement pas la plus efficace. Il faut avouer que le coude n'est guère prononcé. Résultat, nous hésitons toujours entre 3 et 5 classes.

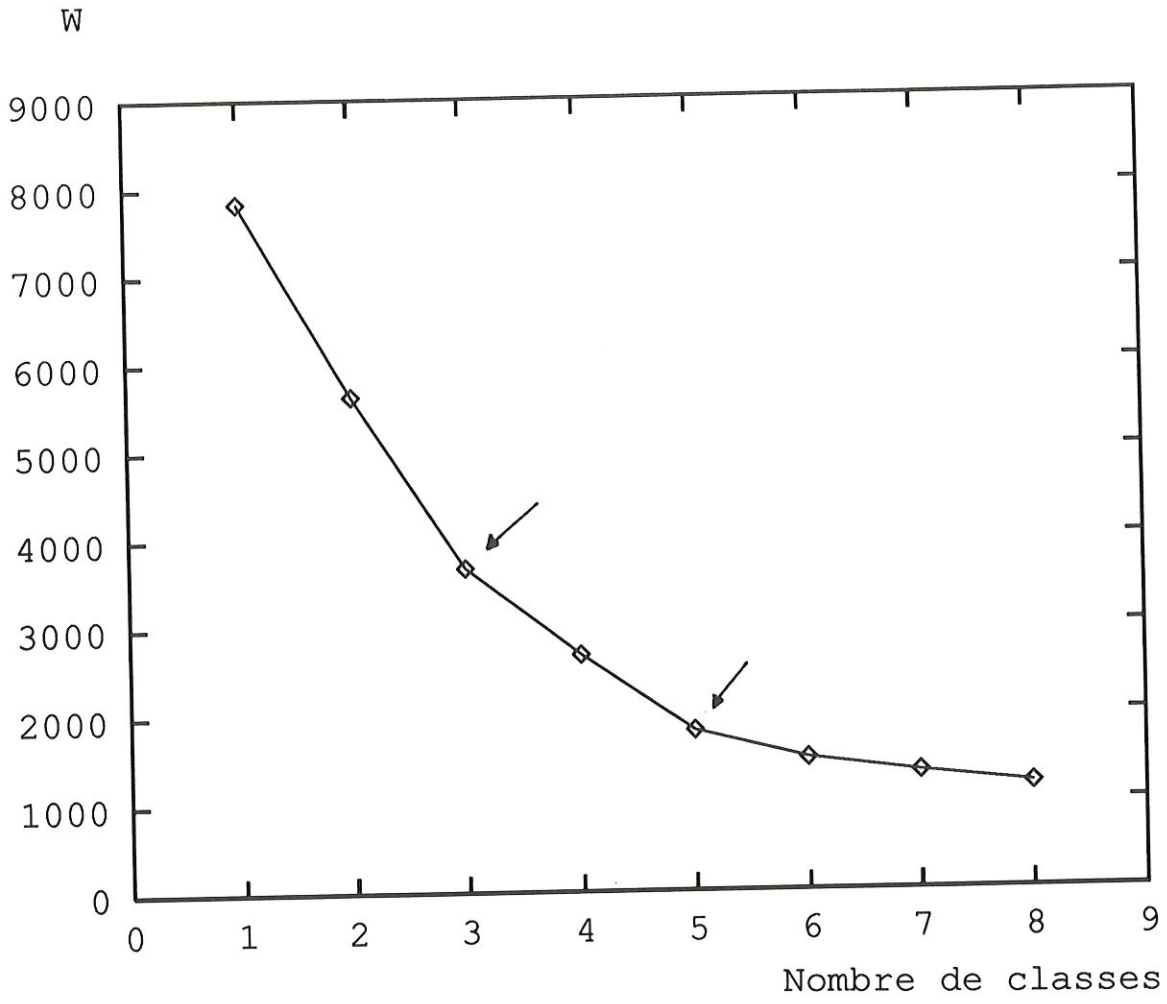


*FIG.*1.3.

Dès lors, nous pouvons dire que le problème reste entier. Nous aurons l'occasion d'y revenir par la suite.

1.3 Problème du nombre de classes

1.3.1 Exemple illustratif

Prenons un exemple tiré d'un contexte familier pour illustrer ce problème: Un jeu de 52 cartes à répartir en plusieurs classes.

Il apparaît d'emblée que le choix est difficile parce que plusieurs groupements possibles nous viennent à l'esprit.

Voici (FIG.1.4), les quatre possibilités les plus souvent citées :

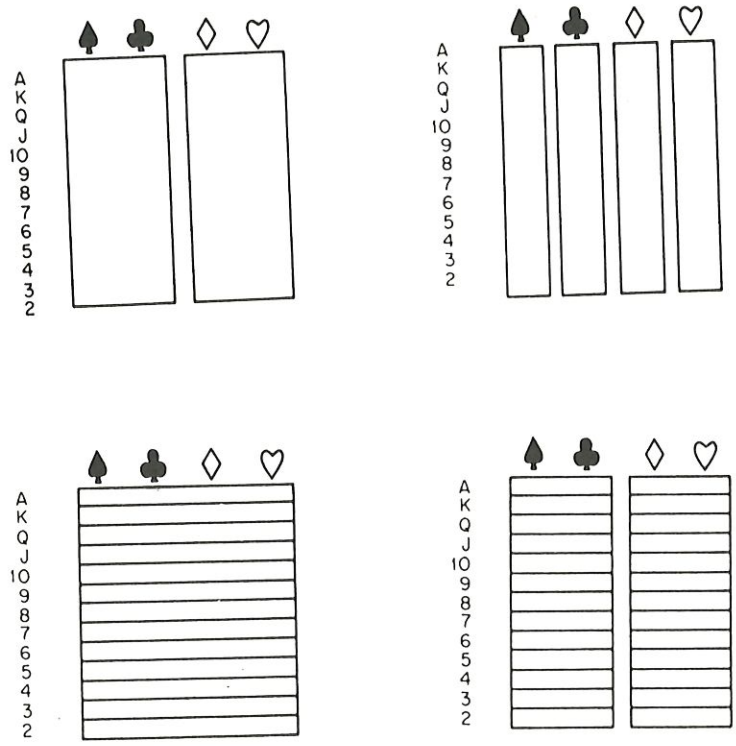


*FIG.*1.4

Nous le voyons, ces différents classements sont tous plausibles et sensés. C'est d'ailleurs le cas pour n'importe quel ensemble de données. Il ne faut pas nécessairement chercher LA seule bonne classification. Le choix entre deux classifications dépend parfois d'autres considérations que mathématiques.

1.3.2 Ebauche de solutions

Un problème pratique substantiel lors d'une classification automatique est de déterminer le nombre de classes dans l'ensemble des données. Certaines méthodes donnent une configuration pour chaque nombre de classes allant de *un* (la totalité de l'ensemble de données) à *n*, le nombre d'individus (chaque classe contient un seul membre). D'autres algorithmes trouvent la meilleure structure pour un nombre donné de classes.

Une des réponses possibles, c'est d'appliquer classiquement la méthode du coude décrite précédemment mais nous l'avons vu, ce n'est pas toujours possible.

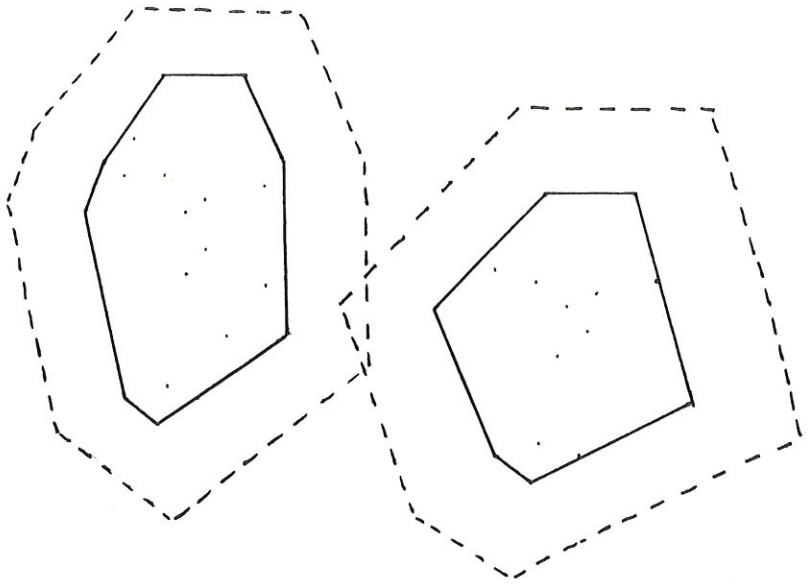
D'autres voies s'offrent à nous comme par exemple, l'approche par estimation de convexes [7] :

Intuitivement, lorsque nous avons k domaines convexes disjoints constitués par k classes d'une partition optimale des données, nous pouvons estimer les domaines en les dilatant à partir de leur centre de gravité (FIG. 1.5).

Enveloppe convexe. ———
Enveloppe dilatée. - - - - -

*FIG.*1.5

Si ces k ensembles dilatés sont disjoints, nous pouvons accepter l'hypothèse que la partition en k classes est optimale, à condition que l'on ait déjà accepté les hypothèses selon lesquelles la partition des données comprend 2,...,(k-1) classes (FIG. 1.6).



Enveloppe convexe. ————
Enveloppe dilatée. - - - - - - - -

*FIG.*1.6

Dans cette conception, on voit apparaître une approche statistique : les tests d'hypothèse. C'est la raison pour laquelle dans le chapitre suivant, nous allons aborder le problème de la détermination du nombre de classes par la théorie des tests d'hypothèse mais, dans un autre contexte que celui-ci.

Chapitre 2

Approche par les tests d'hypothèse

2.1 Introduction

C'est A. Hardy [7] qui évoque pour la première fois l'idée d'utiliser la méthode des hypervolumes pour déterminer le nombre de classes naturelles dans les données par le test d'hypothèse.

Comme il le souligne, l'hypothèse nulle peut exprimer différentes choses, entre autres :

- L'absence de stuctures dans les données.
- L'absence d'une structure particulière.
- La présence de k classes naturelles.
- etc.

Il propose le test suivant :

Soit $X_1, X_2, \ldots, X_n$ un échantillon aléatoire d'un processus de Poisson dans t domaines convexes disjoints d'un espace euclidien à DIM dimensions.
Soit le test d'hypothèse :

$$H_o : t = k$$

contre

$$H_1 : t = k - 1$$

Désignons par :

$C = \{C_1, C_2, \ldots, C_k\}$ la partition optimale en k classes.

$D = \{D_1, D_2, \ldots, D_{k-1}\}$ la partition optimale en k-1 classes.

Les paramètres C et D sont des paramètres de nuisance, de dimension infinie.
Le quotient de vraisemblance Q s'écrit sous la forme :

$$\begin{aligned} Q(\tilde{x}) &= (sup_D f_D(\tilde{x} : t = k-1))/(sup_C f_C(\tilde{x} : t = k)) \\ &= [1/(m(C))^n]/[1/(m(D))^n] \\ &= [W(P,k)/W(P,k-1)]^n \end{aligned}$$

La région critique sera de la forme :

$$\begin{aligned} \text{RC} &= \{Q(\tilde{x}) > c\} \\ &= \{W(P,k)/W(P,k-1) > c'\} \\ &= \{S > c'\} \end{aligned}$$

Remarquons que $0 \leq S \leq 1$.

Nous pouvons utiliser la règle de décision suivante :

On rejette H_0 en faveur de H_1 si $\frac{W(P,k)}{W(P,k-1)}$ prend de trop grandes valeurs, c'est-à-dire si S est proche de 1. Malheureusement, nous ne connaissons pas la distribution de la statistique S. Il ne nous est donc pas possible de déterminer le niveau d'incertitude avec lequel on accepte ou on rejette une hypothèse.

Mais avant de détailler les hypothèses que nous avons testées dans ce mémoire, voyons d'abord le modèle théorique sur lequel se base notre recherche.

2.2 Processus de Poisson

Le processus de Poisson est un modèle pour des points distribués aléatoirement et indépendamment dans l'espace.

De manière formelle :

Soit E un espace localement compact à base dénombrable.

Soit ϵ la tribu borélienne de E.

Soit N(A) le nombre de points dans A.

Dans la suite de notre mémoire, nous allons prendre E = $\mathbb{R}^{DIM}$ et ϵ la tribu des boréliens de $\mathbb{R}^{DIM}$.

Définition :

Soit (E,ϵ) un espace mesurable. Un processus ponctuel N sur E est un **processus de Poisson de moyenne μ** si :

a. $A_1, \ldots, A_n$ est une famille de boréliens de E disjoints deux à deux alors $N(A_1), \ldots, N(A_n)$ sont des variables aléatoires indépendantes ;

b. Pour tout borélien A de E et pour tout $k \geq 0$,

$$P\{N(A) = k\} = e^{-\mu(A)} \frac{\mu(A)^k}{k!}$$

Par convention N(A) = ∞ presque sûrement si $\mu(A) = \infty$

Ainsi, N a des accroissement indépendants et pour chaque borélien A de E, N(A) suit une loi de Poisson de moyenne $\mu(A)$.

Le processus de Poisson est dit ordinaire, homogène ou stationnaire sur $\mathbb{R}^+$ si $\mu(A) = \lambda m(A)$, où m(A) est la mesure de Lebesgue de A et $\lambda \in]0, +\infty[$ est une constante appelé l'intensité ou le taux du processus N. Les autres processus de Poisson sur $\mathbb{R}^+$ sont appelés processus de Poisson non homogènes.

Une propriété importante caractérise ce modèle:

Conditionnellement au fait qu'il y a n points dans A, ceux-ci sont distribués uniformément et indépendamment dans A.

L'idée d'indépendance et ses résultats mathématiques attrayants en font un concept populaire auprès des probabilistes. Le processus de Poisson homogène est quelquefois appelé processus de Poisson complètement aléatoire.

Nous pouvons avoir un processus de Poisson dans un espace de n'importe quelle dimension ($\mathbb{R}, \mathbb{R}^2, \mathbb{R}^3, \ldots$).

Pour clarifier, considérons l'espace à 2 dimensions. Il suffit alors de lire le mot "aire" à la place du mot "volume". Les modifications sont donc très simples.

Dans notre étude, nous considérons donc que nous traitons un problème de classification lorsque les points que nous observons sont engendrés par un processus de Poisson homogène et sont distribués dans k domaines disjoints $(C_i)_{1 \leq i \leq k}$ que nous voulons retrouver.

Nous avons ainsi toutes les bases théoriques pour faire notre test d'hypothèse.

2.3 Le test d'hypothèse

La méthode de V. Gendarme [5] dans sa phase de recollement passe de v classes (obtenues après division) à k classes (la partition optimale du problème de classification) où k et v sont fixés et donnés par l'utilisateur.

Pendant cette phase de recollement, dans un souci de ne plus devoir donner un nombre k a priori, nous proposons le test d'hypothèse suivant :

> Soit D_1 un domaine convexe contenant n_1 points et $m_1 = m(D_1)$ la mesure de Lebesgue de l'enveloppe convexe (p.ex. en dimension 2, c'est l'aire)
> et D_2 un autre domaine convexe disjoint de D_1, contenant n_2 points et $m_2 = m(D_2)$.
> Nous noterons :

- le nombre total de points $n = n_1 + n_2$
- D le domaine convexe qui contient ces n points
- $m = m(D)$

Le paramètre auquel nous nous intéressons c'est m, il est inconnu. Les paramètres n_1, n_2 sont connus et nous considérons m_1, m_2 comme des paramètres de nuisance qu'il faudra estimer. Mais nous y reviendrons dans la suite.

A chaque étape, le problème est de savoir s'il faut regrouper deux classes ou les laisser séparées. Les hypothèses seront ainsi :

H_0 : Les n_1 et n_2 points sont distribués dans deux intervalles disjoints.
contre
H_1 : Les n points sont distribués dans un même intervalle.

En choisissant les hypothèses dans cet ordre, nous empêchons le regroupement de deux classes qui doivent être séparées.

Le quotient de vraisemblance Q s'écrit sous la forme :

$$Q(\tilde{x}) = \frac{f_0(\tilde{x})}{f_1(\tilde{x})}$$

où $f_0(\tilde{x})$ est la vraisemblance du modèle sous H_0

$$f_0(\tilde{x}) = \frac{1}{m_1^{n_1}} \frac{1}{m_2^{n_2}}$$

et $f_1(\tilde{x})$ est la vraisemblance du modèle sous H_1

$$f_1(\tilde{x}) = \frac{1}{m^{n_1+n_2}}$$

Le quotient de vraisemblance dans ce cas vaut:

$$Q(\tilde{x}) = \frac{m^n}{m_1^{n_1} m_2^{n_2}}$$

La région critique vaut :

$$RC = \{\tilde{x} \text{ t.q. } Q(\tilde{x}) \leq c\}$$

Nous obtenons alors :

$$\frac{m^n}{m_1^{n_1} m_2^{n_2}} \leq c$$

Comme m_1, m_2, n_1, n_2 sont des constantes données :

$$m^n \leq c m_1^{n_1} m_2^{n_2} = c'$$

En passant au logarithme, nous avons :

$$n \log m \leq \log c' = c'' \text{ ou encore } \log m \leq \frac{c''}{n} = c'''$$

En repassant à l'exponentielle, nous avons finalement :

$$m \leq e^{c'''} = c'^{v}$$

Si nous notons m' l'ecart entre les classes 1 et 2 i.e. $m' = m - (m_1 + m_2)$
alors nous obtenons :

$$m' \leq c'^{v} - (m_1 + m_2) = c^{v} = K_\alpha.$$

Notre région critique devient :

$$RC = \{\tilde{x} \text{ tq. } m' \leq K_\alpha\}$$

Si nous notons α, l'erreur de première espèce c'est-à-dire la probabilité sous H_0 d'être dans la région critique appelée aussi le niveau du test,

nous avons :

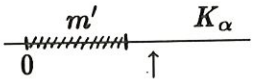
$$P(m' \leq K_\alpha) = \alpha$$

D'autre part,

$$P(m' \leq K_\alpha) = P(N(\tilde{K}_\alpha) > 0)$$

où $N(\tilde{K}_\alpha)$ est le nombre de points dans $\tilde{K}_\alpha$.

En effet,dans $\mathbb{R}^+$, nous pouvons illustrer cela par ce petit dessin :



$\tilde{K}_\alpha$ peut être par exemple une partie de $\mathbb{R}^{DIM}$

L'événement "l'espace vide est plus petit que $\tilde{K}_\alpha$" est équivalent à "il y a au moins un point dans $\tilde{K}_\alpha$" (situé dans le complémentaire de m' dans K_α).

ou encore

$$P(m' \leq K_\alpha) = 1 - P(N(\tilde{K}_\alpha) = 0)$$

Or nous avons vu au paragraphe 2.2 que

$$P(N(\widetilde{K_\alpha}) = 0) = e^{-\lambda K_\alpha}$$

d'où

$$\alpha = 1 - e^{-\lambda K_\alpha}$$

ce qui est équivalent à

$$K_\alpha = -\frac{\ln(1-\alpha)}{\lambda}$$

La stratégie du test ou la règle de décision est la suivante :

Si l'espace vide entre les deux classes est trop grand i.e. $m' \geq K_\alpha$,
nous acceptons H_0 et nous arrêtons le recollement.
Au contraire, si l'espace vide est suffisamment petit i.e. $m' < K_\alpha$,
nous rejetons H_0 au profit de H_1 et nous recollons les deux classes.

Dans le chapitre suivant, nous allons voir les problèmes théoriques et les difficultés pratiques de ce test.

Chapitre 3

Remarques et difficultés du test

3.1 Le point de vue théorique

3.1.1 Le niveau α du test

On définit le niveau du test par :

$$P_{H_0}(RégionCritique) \leq \alpha$$

dans notre cas, nous obtenons

$$1 - e^{-\lambda K_\alpha} \leq \alpha$$

$$\Leftrightarrow$$

$$K_\alpha \leq -\frac{\ln(1-\alpha)}{\lambda}$$

Et le test est implémenté de la manière suivante :

Si $K_\alpha \leq m'$
alors, nous arrêtons de recoller .
où nous calculons le K_α avec l'égalité.

Autrement dit, nous donnons une valeur par excès de K_α. Ce qui nous conduit à dire que le niveau du test demandé à l'utilisateur pour calculer le K_α, n'est pas le niveau réel du test. Par exemple, si nous donnons à α la valeur 0.01, en pratique la probabilité de recoller deux classes naturellement disjointes est plus grande qu'une chance sur 100.

3.1.2 L'estimation de l'intensité du processus

Lorsque nous avons à faire à un processus homogène de Poisson, l'espérance du nombre de points dans un ensemble A est proportionnel à la mesure de Lebesgue de cet ensemble. Le facteur de proportionnalité est appelé intensité du processus et nous le notons λ i.e. $E[N(A)] = \lambda m(A)$

λ est une constante strictement positive.

Lorsque nous estimons ce paramètre λ , nous adopterons le point de vue de A.F. Karr [9]. Le modèle statistique est un processus de Poisson sur $\mathbb{R}^+$ où l'intensité du processus λ est inconnue.
Soit la famille de lois de probabilités suivante :

$$P = \{P_\lambda : \lambda \in (o, \infty)\}$$

Proposition 1

Soient $X_1, X_2, \ldots, X_n$ des v.a. i.i.d. qui obéïssent à la loi exponentielle de moyenne $\theta = \frac{1}{\lambda}$.
C'est-à-dire

$$f(x, \theta) = \frac{1}{\theta} e^{-x/\theta} \ , \ \theta \in (0, +\infty)$$

a) L'estimateur du maximum de vraisemblance de λ est

$$\hat{\lambda} = \frac{1}{\hat{\theta}} \text{ où } \hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

b) Sous P_λ et pour chaque λ, $\hat{\lambda} \longrightarrow \lambda$ (presque sûrement).

c) Sous P_λ, $\sqrt{n}(\hat{\lambda} - \lambda) \xrightarrow{d} N(0, \lambda^2)$.

Proposition2

Soit

$$\hat{\lambda} = \frac{N(A)}{m(A)}$$

Sous P_λ, nous avons les résultats suivants :

a) $\hat{\lambda} \longrightarrow \lambda$ presque sûrement.

b) $\sqrt{m(A)}(\hat{\lambda} - \lambda) \xrightarrow{d} N(0, \lambda)$.

Cette statistique est exhaustive.

Nous utiliserons dans la suite l'estimateur donné par la proposition 2. $\hat{\lambda}$ est alors une densité de points.

D'autre part, l'estimation de λ lorsque nous faisons un test d'hypothèse, se fait sous H_0.

Enfin, ce test se déroule dans un environnement particulier c'est-à-dire qu'il y a d'autres classes qui entrent en jeu.

Prenons un ensemble de 4 classes (FIG 3.1):
Chaque classe i ($i = 1, \ldots, 4$) contient n_i points et couvre une surface m_i.

Nous définissons VOL_{tot} comme la surface de l'enveloppe convexe de tous les NP_{tot} points.

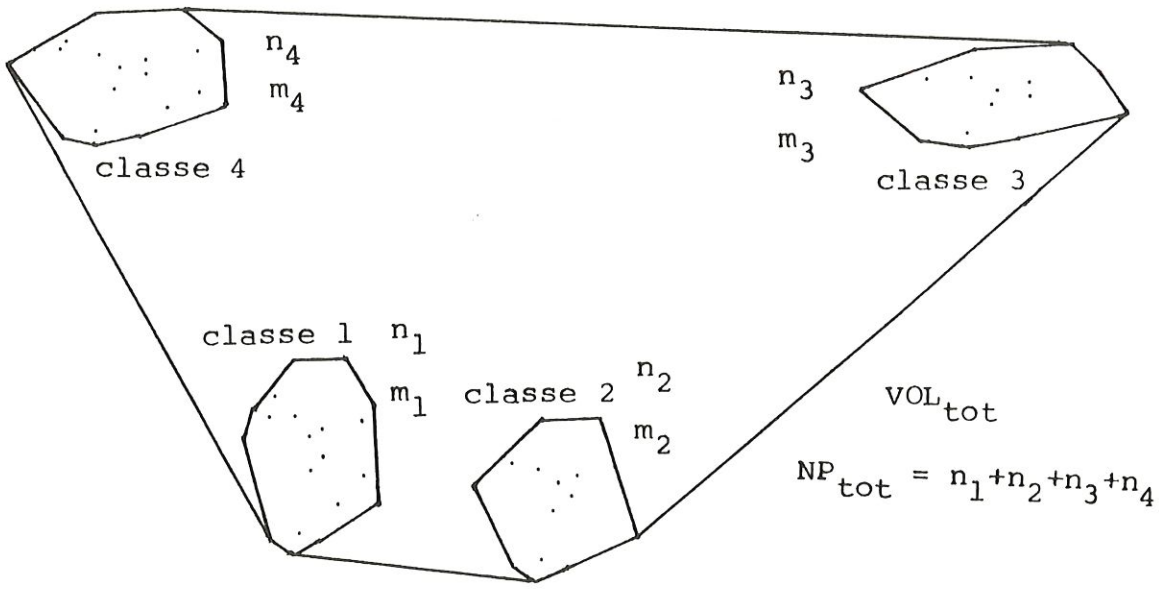


*FIG.*3.1.

Compte tenu des considérations énoncées ci-dessus, nous constatons qu'il y a deux manières d'estimer le λ.

Pour l'ensemble des données avec un processus homogène, il faudrait prendre pour estimation de λ :

$$\hat{\lambda} = \frac{NP_{tot}}{VOL_{tot}}$$

Si nous tenons compte de l'hypothèse nulle H_0 : " Les deux classes sont disjointes ", nous devons alors estimer le paramètre λ par :

$$\hat{\lambda} = \frac{n_1 + n_2}{m_1 + m_2}$$

Mais ici, $\hat{\lambda}$ n'est plus du tout constant et le processus n'est plus homogène . Il varie proportionnellement au nombre de points dans l'intervalle et est inversément proportionnel à la somme des surfaces des deux domaines. De plus, au niveau pratique, il arrive parfois que m_1 et m_2 soient

nuls. Le test n'est donc pas appliquable pour des problèmes où il y a des classes de mesures nulles.

Dans le cas où il y a uniquement deux classes, le premier estimateur devient :

$$\hat{\lambda} = \frac{n_1 + n_2}{VOL_{tot}}$$

Et comme nous l'avons vu en 2.2, il a de très bonnes propriétés théoriques. C'est donc cet estimateur que nous allons utiliser dans la suite.

3.2 Le point de vue pratique

3.2.1 Implémentation du test

Le test peut être implémenté sur n'importe quel algorithme qui comporte une phase de recollement, de deux classes à la fois.

Dans le programme par maximisation, nous avions (FIG 3.2) :

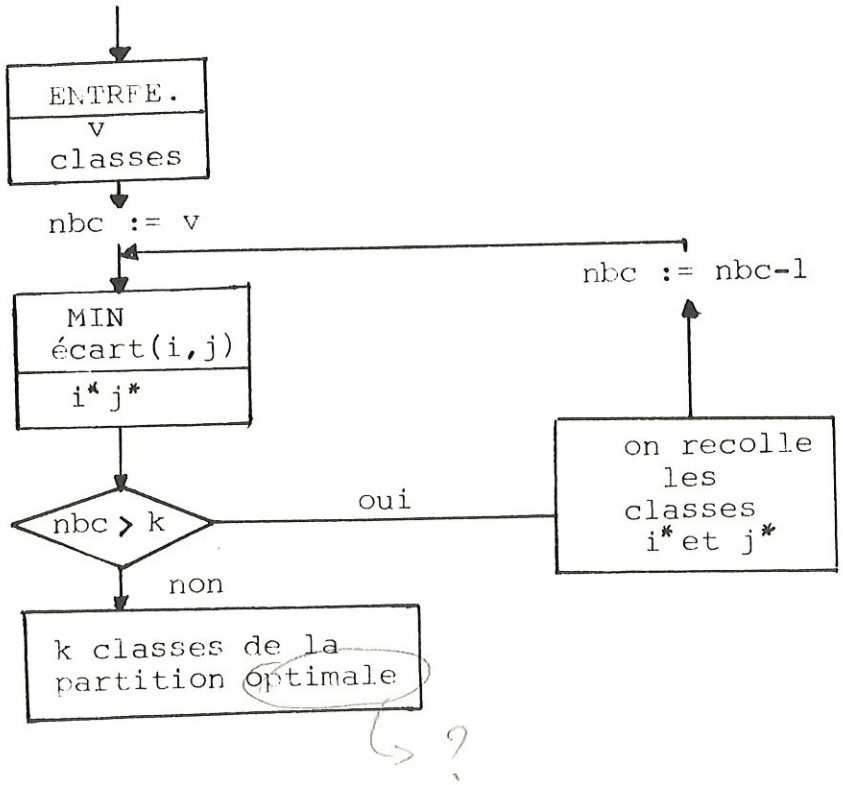


FIG.3.2.

Voici comment nous appliquons le test sur ce programme (FIG. 3.3).

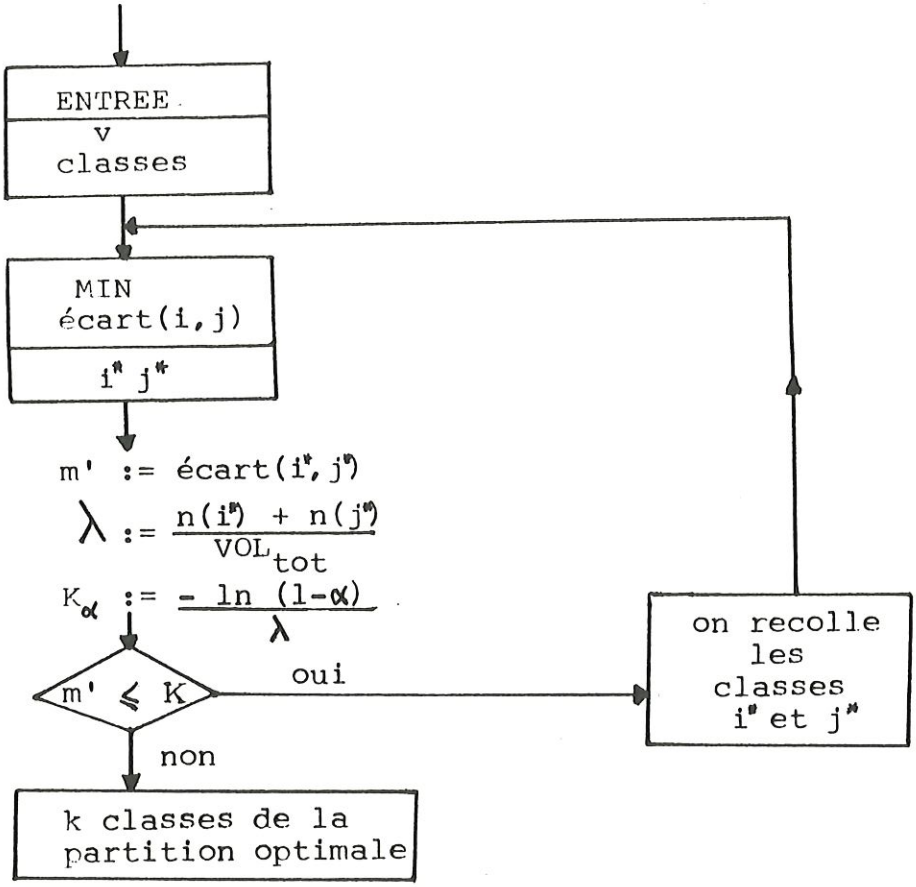


FIG.3.3.

Nous le voyons, l'algorithme implémenté est tributaire de l'algorithme de classification sur lequel il est greffé. Comme le test n'intervient que pour la phase de recollement, nous pouvons dire que d'une certaine manière, il n'améliore pas la classification pour l'affectation des points. Prenons un exemple. Si nous avons un ensemble de données constitués de 4 classes, et si l'algorithme de classification retrouve effectivement ces classes , deux cas peuvent être envisagés lorsque nous ajoutons le test d'hypothèse.

- Soit nous retrouvons effectivement les 4 classes de départ et dans ce cas, la classification est identique.

- Soit nous trouvons par exemple 3 ou 5 classes et la classification est alors différente mais malgré tout, le partitionnement en 4 classes reste le partitionnement intermédiaire entre 3 et 5 classes.

3.2.2 Choix du seuil α

Le choix du seuil du test d'hypothèse α n'est pas univoque. Mais au vu de l'expérimentation, il se situe entre $\alpha = 0.05$ et $\alpha = 0.1$, ce qui est une fourchette très satisfaisante. Cette fourchette est d'ailleurs utilisée couramment dans les test d'hypothèse.

Il est donc raisonnable d'y prendre plusieurs valeurs de α et de comparer les résultats obtenus. Cette comparaison ne doit pas se faire uniquement sur le nombre de classes obtenues après recollement mais aussi sur les résultats K_α et m' = écart (i,j) qui sont deux variables importantes du test. En effet, nous recollons tant que K_α est plus grand que l'écart entre les deux classes considérées. Parfois, il ne faudrait pas "grand chose" pour que le recollement puisse continuer (voir l'exemple de GAVAERT). Dans ce cas, nous suggérons d'augmenter légèrement la valeur de α.

Remarquons enfin que, plus α augmente, plus K_α augmente, c'est-à-dire que le recollement est plus fréquent et donc il y a une tendance à obtenir un petit nombre de classes.

3.2.3 Les exemples générés

Pour les données pratiques qui nous parviennent, nous supposons avoir à faire à un processus de Poisson homogène mais en réalité, cette hypothèse mérite d'être vérifiée.

Nous utiliserons lors de la génération des points obéïssant aux processus de Poisson, l'algorithme connu et repris par Ripley [11]. Le lecteur intéressé par ce sujet, trouvera en annexe de notre mémoire, le programme utilisé pour la génération de nos points.

3.2.4 Le nombre v de classes

Nous l'avons vu au paragraphe 1.2.2, v est le nombre de classes après division. C'est une constante fixée . Dans le cas qui nous préoccupe, nous avons fixé la constante à dix et nous devons recoller ces classes (deux à la fois) jusqu'à l'obtention de k classes via le test d'hypothèse.

La détermination de cette constante à une influence sur le recollement car comme nous l'avons déjà souligné, le test est conçu de sorte qu'il ne recolle pas deux classes qui doivent être séparées. Par conséquent, l'algorithme pour la phase de recollement peut s'arrêter plus rapidement si le nombre de classes à recoller est plus grand et inversément.

Voici quelques exemples :

- Pour deux classes éloignées.

  v=10

$$\alpha = 0.1 \longrightarrow 2classes.$$

$$\alpha = 0.05 \longrightarrow 2classes.$$

$$\alpha = 0.01 \longrightarrow 8classses.$$

  v=7

$$\alpha = 0.1 \longrightarrow 2classes.$$

$$\alpha = 0.05 \longrightarrow 2classes.$$

$$\alpha = 0.01 \longrightarrow 5classes.$$

  v=13

$$\alpha = 0.1 \longrightarrow 2classes.$$

$$\alpha = 0.05 \longrightarrow 2classes.$$

$$\alpha = 0.01 \longrightarrow 8classes.$$

- Pour deux classes proches.

  v=10

$$\alpha = 0.1 \longrightarrow 2classes.$$

$$\alpha = 0.05 \longrightarrow 2classes.$$

$$\alpha = 0.01 \longrightarrow 10classses.$$

  v=7

$$\alpha = 0.1 \longrightarrow 2classes.$$

$$\alpha = 0.05 \longrightarrow 2 classes.$$

$$\alpha = 0.01 \longrightarrow 7 classes.$$

v=13

$$\alpha = 0.1 \longrightarrow 2 classes.$$

$$\alpha = 0.05 \longrightarrow 2 classes.$$

$$\alpha = 0.01 \longrightarrow 12 classes.$$

- Pour trois classes rectangulaires.

v=10

$$\alpha = 0.1 \longrightarrow 3 classes.$$

$$\alpha = 0.05 \longrightarrow 9 classes.$$

$$\alpha = 0.01 \longrightarrow 10 classses.$$

v=7

$$\alpha = 0.1 \longrightarrow 3 classes.$$

$$\alpha = 0.05 \longrightarrow 4 classes.$$

$$\alpha = 0.01 \longrightarrow 7 classes.$$

v=13

$$\alpha = 0.1 \longrightarrow 3 classes.$$

$$\alpha = 0.05 \longrightarrow 9 classes.$$

$$\alpha = 0.01 \longrightarrow 13 classes.$$

Nous remarquons que la détermination de v est difficile car elle est partagée entre deux considérations opposées.

Si nous prenons v trop petit, nous risquons de diviser l'ensemble des données en trop peu de classes et d'être en dessous du nombre de classes de la partition optimale avant même de recoller.

De plus, v doit être suffisamment grand pour pouvoir recoller des classes malencontreusement séparées lors de la division.

D'autre part, si v est trop grand, il y a un risque d'arrêter la phase de recollement avant l'obtention du nombre de classes de la partition optimale.

Constatons enfin que la différence entre les recollements pour différentes valeurs de v est plus nette quand α diminue. Au seuil $\alpha = 0.1$, il y a apparemment aucun problème pour les exemples traités.

Il est grand temps, à présent d'illustrer notre propos.

Chapitre 4

Applications

Dans ce chapitre, nous allons analyser plusieurs exemples et comparer les résultats obtenus par le test d'hypothèse et par la méthode du coude.

4.1 Les données générées

Les différents ensembles de données que nous traitons dans ce paragraphe sont constitués d'un certain nombre de points générés aléatoirement dans k domaines convexes disjoints du plan (Voir le programme en annexe).

Plusieurs cas de figures très simples ont été analysés.

- Pour deux classes carrées de 100 points chacune, très éloignées (FIG. 4.1), nous obtenons, pour un niveau α de 0.1, une partition optimale de deux classes. Ce résultat est encore obtenu pour α valant 0.05.

*FIG.*4.1

- Si nous rapprochons ces deux classes (FIG. 4.2), nous avons toujours le même résultat.

*FIG.*4.2

- Pour deux classes proches et très allongées de 100 points chacune (FIG. 4.3), l'algorithme recolle jusque deux classes pour le niveau $\alpha = 0.1$ et s'arrête à quatre classes pour α valant 0.05.
  Nous pouvons être très satisfaits de ce résultat car lorsque les classes sont très étirées, l'espace vide entre deux points d'une même classe peut être plus grand que ce que nous imaginons et dans ce cas, nous obtenons quatre classes.

*FIG.*4.3

- Pour deux classes contenant chacune 100 points mais dont la surface de la première est 12 fois plus grande que la seconde (FIG. 4.4), nous obtenons encore deux classes pour une fourchette de α comprise entre 0.1 et 0.05.

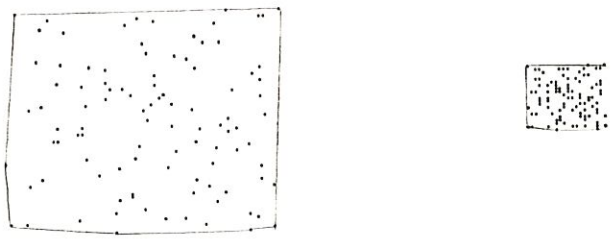


*FIG.*4.4

- Prenons le cas de deux classes de même superficie mais dont l'une contient trois fois plus de points que l'autre (FIG. 4.5), nous constatons que les résultats sont toujours aussi bons que précédemment :

$$\alpha = 0.1 \longrightarrow 2classes$$

$$\alpha = 0.05 \longrightarrow 2classes$$

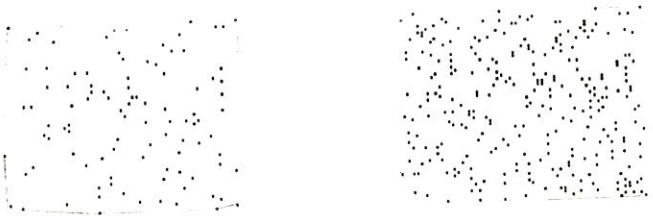


*FIG.*4.5

- Dans le cas de deux classes dont le rapport entre la superficie et le nombre de points est le même c'est-à-dire que le λ du processus de Poisson lors de la génération de l'exemple est identique pour les classes (FIG 4.6), nous constatons que test d'hypothèse fournit encore le nombre de classes de la partition optimale.

$$\alpha = 0.1 \longrightarrow 2classes$$

$$\alpha = 0.05 \longrightarrow 2classes$$

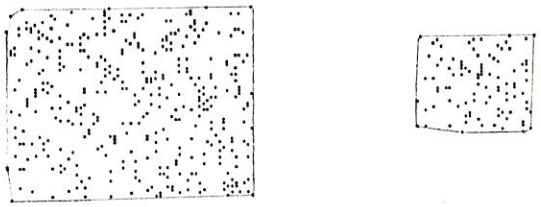


*FIG.*4.6

- Enfin, cet algorithme fournit de très bons résultats pour un nombre de classes supérieur à deux. Nous aurons l'occasion d'y revenir. Voici un exemple simple de trois classes (FIG 4.7). Résultats :

$$\alpha = 0.1 \longrightarrow 3classes$$

$$\alpha = 0.05 \longrightarrow 3classes$$

*FIG.*4.7

Nous venons de passer en revue des exemples classiques. Certes ceux-ci ne sont pas très originaux mais il fallait d'abord s'assurer que l'approche par le test d'hypothèse donne des résultats suffisamment concluants avant de poursuivre les investigations.

Dans la catégorie des domaines convexes, il y a celle des classes sphériques . Ce type de classes n'a pas été traité dans le mémoire de V. Gendarme. C'est pourquoi, soulignons que dans cet exemple (FIG.4.8), l'affectation des points aux classes est excellente.

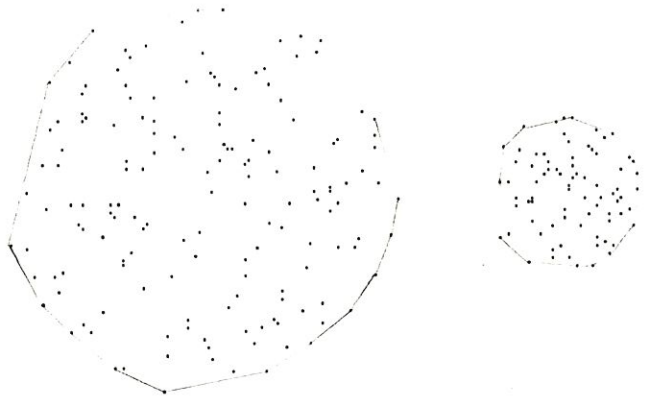


FIG.4.8

Voici les résultats du test :

α	*classes*
0.15	2
0.1	2
0.075	2
0.05	2
0.025	3
0.01	5

Dans l'intervalle habituel de α, nous obtenons les deux classes que nous cherchions. Pour la méthode du coude (FIG. 4.9), nous observons un coude pour la partition en deux classes mais

même si celui-ci est le plus net de la courbe, la valeur du critère W n'est pas stabilisée pour autant puisqu'elle passe de 7929 (2 classes) à 5788 (10 classes).

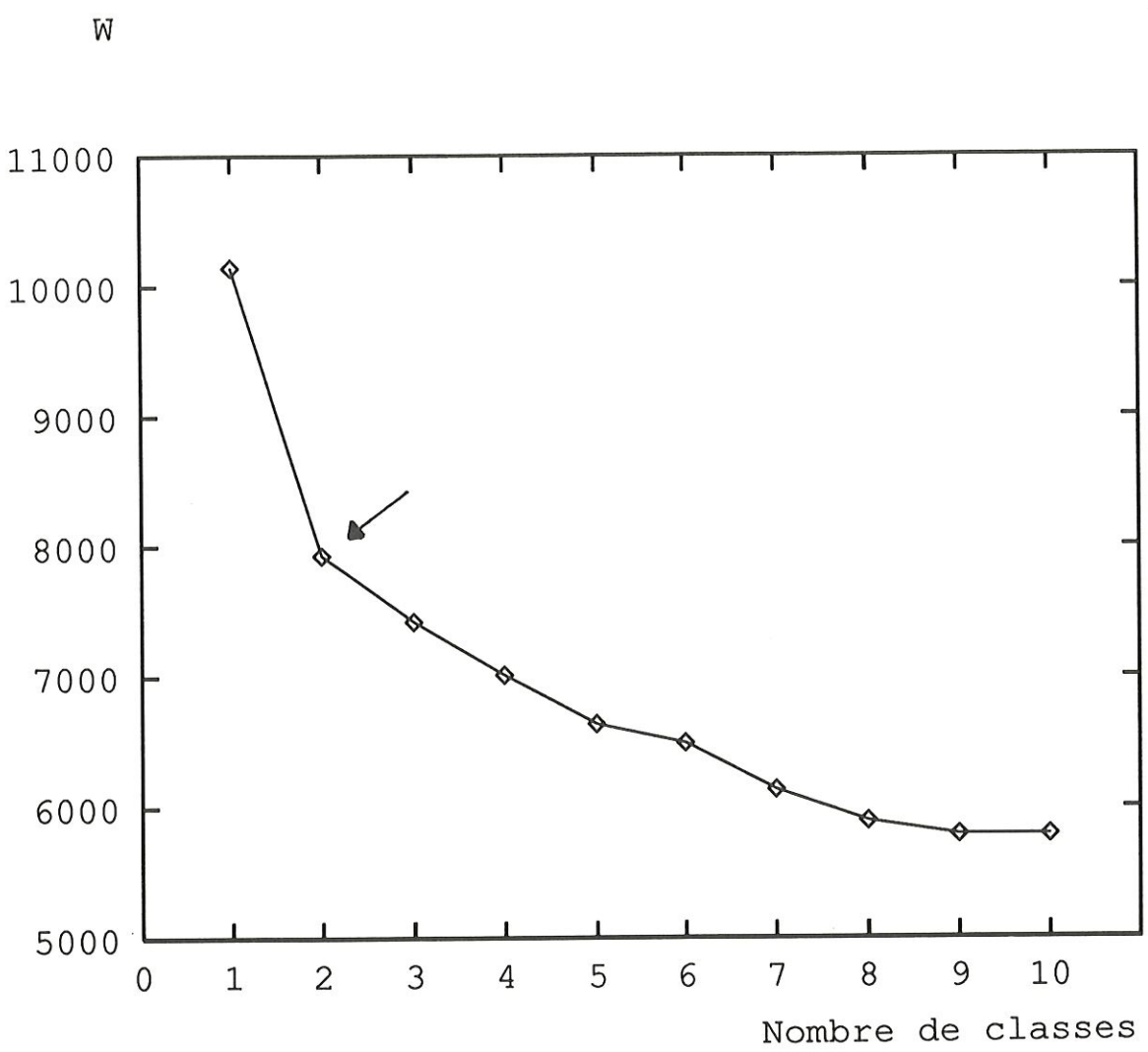


*FIG.*4.9

Voici un autre exemple : 639 points répartis dans quatre domaines convexes du plan. Nous avons trois classes parallèles et une classe oblique. (FIG. 4.10)

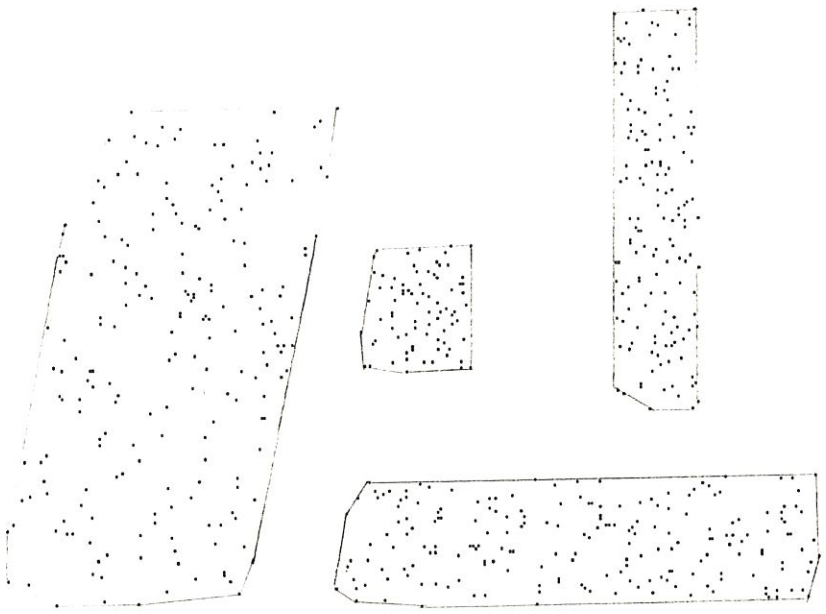


*FIG.*4.10

L'algorithme muni du test d'hypothèse donne les résultats suivants:

α	*classes*
0.1	4
0.08	4
0.075	4
0.06	4
0.05	4
0.04	4
0.02	7
0.01	9

Nous remarquons que les résultats sont concluants. La méthode du coude renforce cette constatation. (FIG. 4.11)

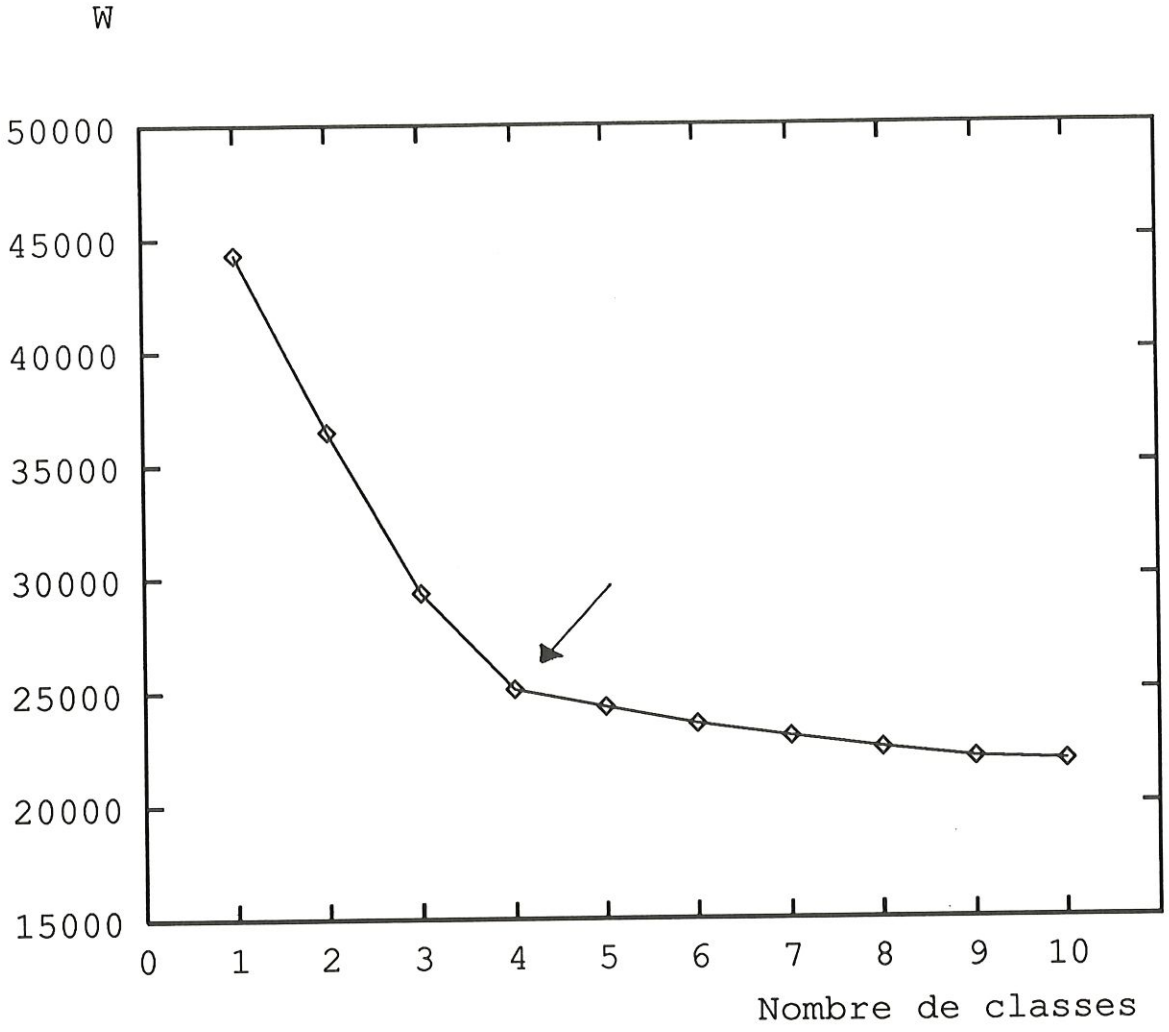


*FIG.*4.11

Le temps C.P.U. demandé pour cette partition en quatre classes est de 23,34 secondes.

Si nous transformons une des classes de l'exemple précédent pour obtenir quatre classes non linéairement séparables (FIG. 4.12), la procédure retrouve le nombre de classes naturelles sans problème au seuil $\alpha = 0.1$ par exemple.

*FIG.*4.12

Tableau :

α	*classes*
0.1	4
0.08	4
0.075	4
0.06	4
0.05	4
0.04	4
0.02	5
0.01	8

Là encore, le coude conforte le résultat. (FIG. 4.13)

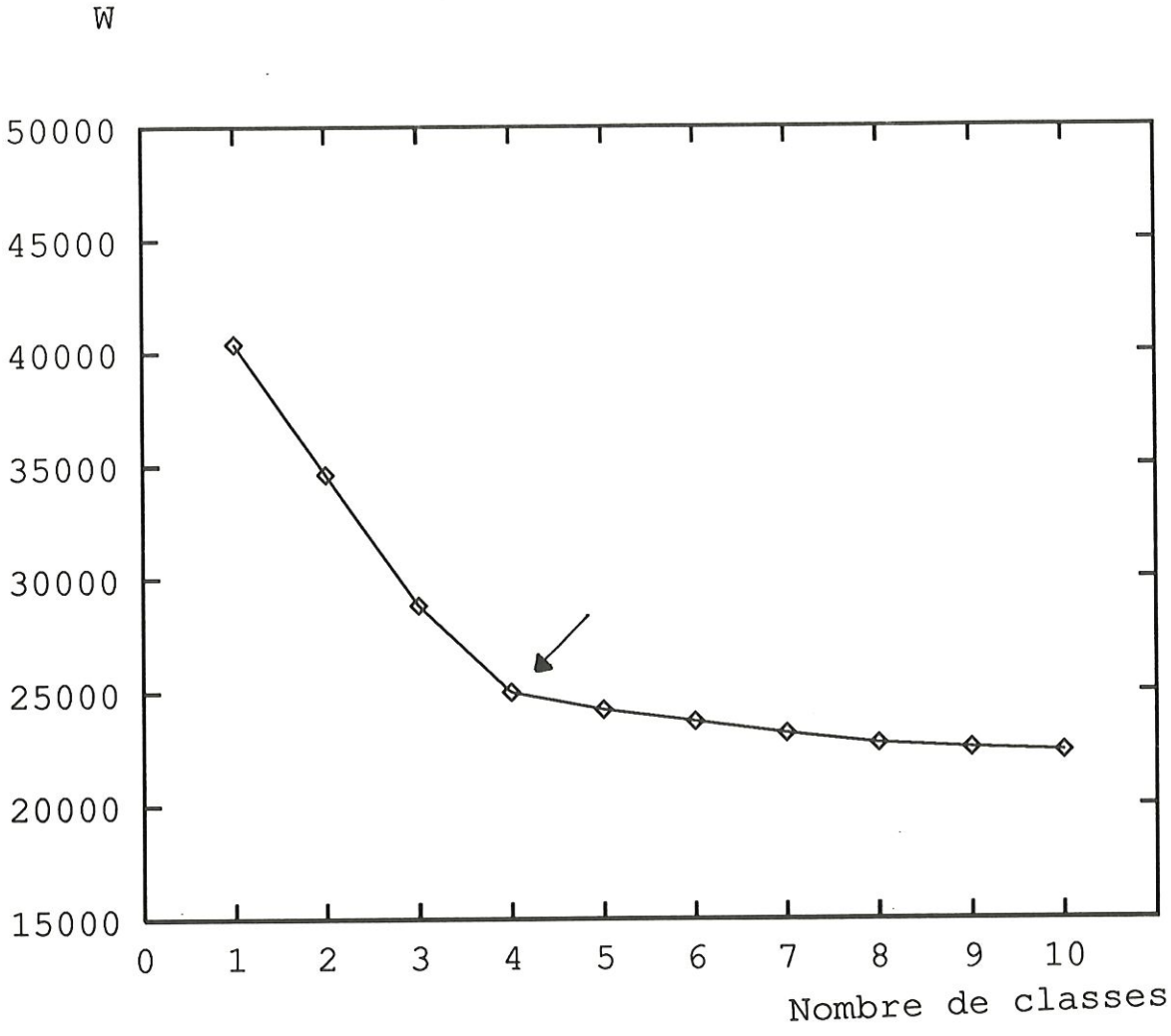


*FIG.*4.13

Cette partition est obtenue après 23,79 secondes.

Etudions à présent un exemple en profondeur :

Considérons, un ensemble de données comportant 750 points répartis dans cinq rectangles. (FIG. 4.14)

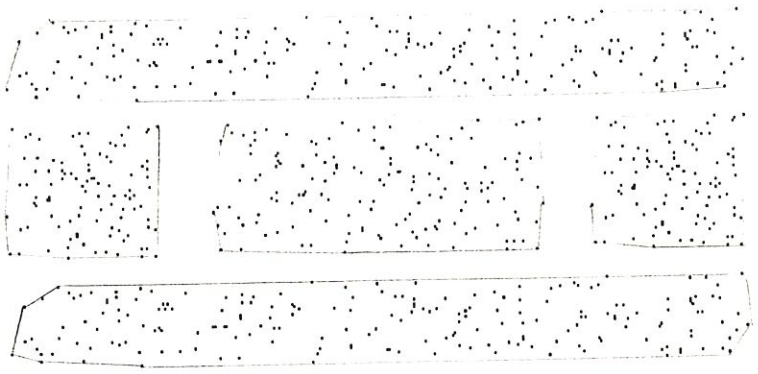


*FIG.*4.14

Voici le tableau qui renseigne le nombre de classes obtenu en fonction du seuil α.

α	*classes*
0.2	2
0.1	3
0.075	3
0.05	3
0.025	5

Si nous prenons la fourchette α habituelle, nous trouvons trois classes. Autrement dit, le test considère que les écarts entre les trois classes du centre ne sont pas suffisamment significatifs. Il faut un seuil de $\alpha = 0.025$ pour arrêter, à cinq classes, le recollement.

Comparons ceci avec la méthode du coude (FIG 4. 15):

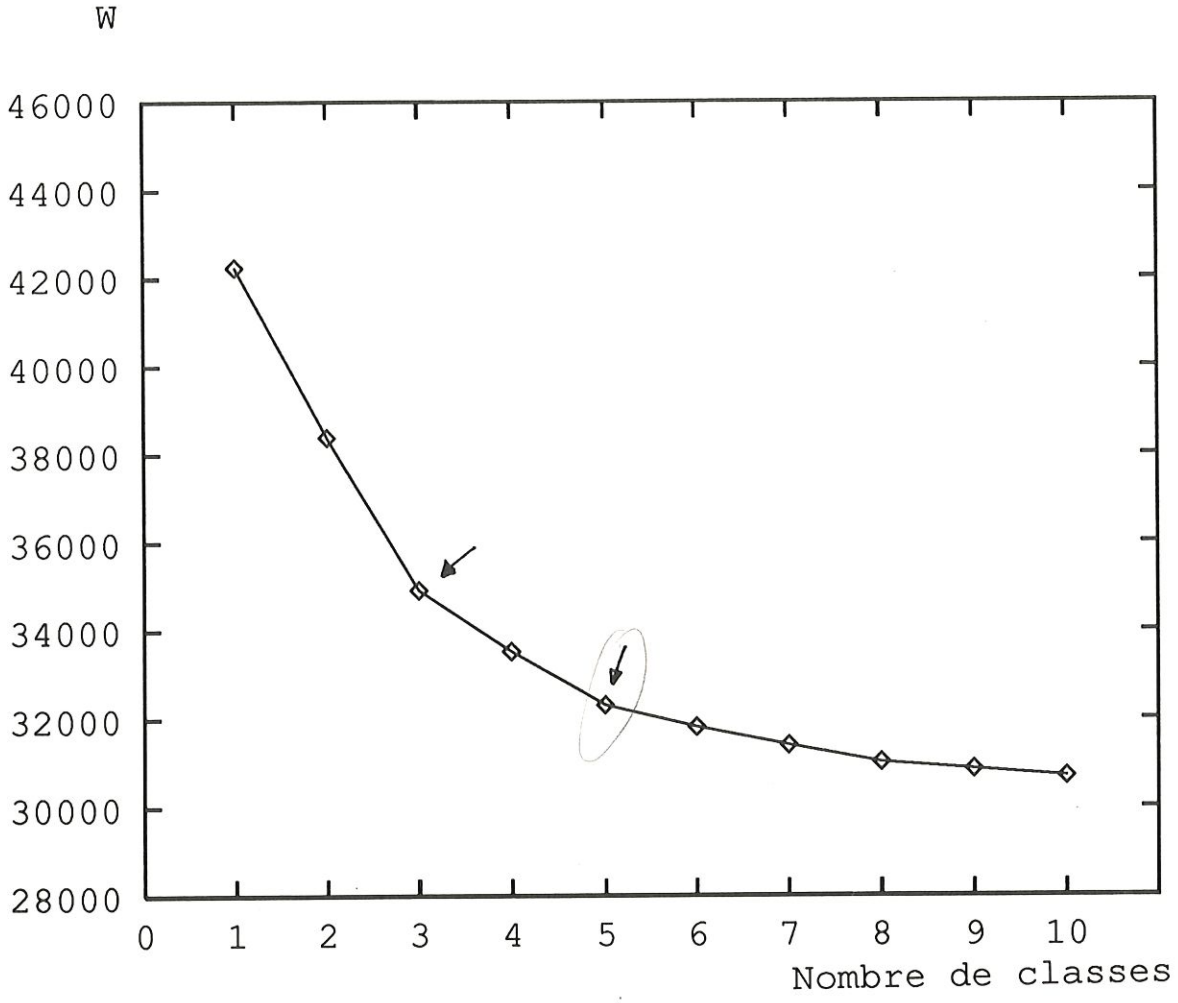
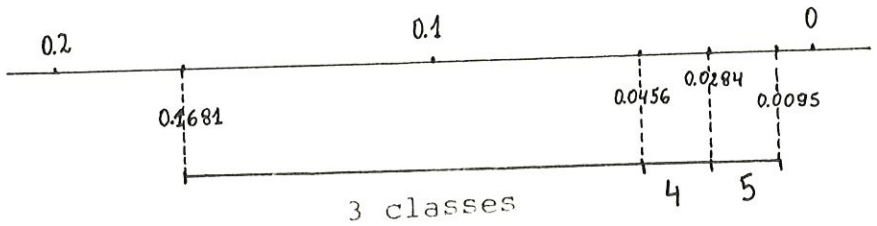


FIG.4.15

Deux coudes sont à observer. Il y a donc, ici aussi, une difficulté à choisir entre un partitionnement en trois ou en cinq classes. Remarquons que ces hésitations sont tout à fait légitimes au vu des données.

Examinons une autre approche :

Dans l'absolu, α peut varier entre 0 et 1 car α est une probabilité. Représentons "les plages de α" pour 3,4,5 classes avec trois chiffres signicatifs.



La plage pour quatre classes est plus petite que celles pour 3 ou 5 classes. Nous pourrions donc dire que plus l'intervalle est étendu, plus les chances d'avoir la bonne partition sont grandes. C'est une voie possible pour déterminer si le nombre de classes que nous obtenons est bon. Nous y reviendrons entre autres, pour l'exemple de Ruspini.

Lorsque nous examinons les valeurs de K_α et la valeur de l'écart entre les classes à recoller, nous obtenons le tableau suivant:

		K_α			
classes	*ecart*	$\alpha = 0.1$	$\alpha = 0.075$	$\alpha = 0.05$	$\alpha = 0.025$
10	156.5	183255	135599	89215	44035
9	156.5	183255	135599	89215	44035
8	395	64678	47858	31487	15542
7	414	5497	4067	2676	1321
6	498	5497	4067	2676	1321
5	1204	4398	3254	2141	1056
4	1394	3141	2354	1529	
3	3493	1999	1479	973	

Reportons ces valeurs sur un graphique.(FIG. 4.16)

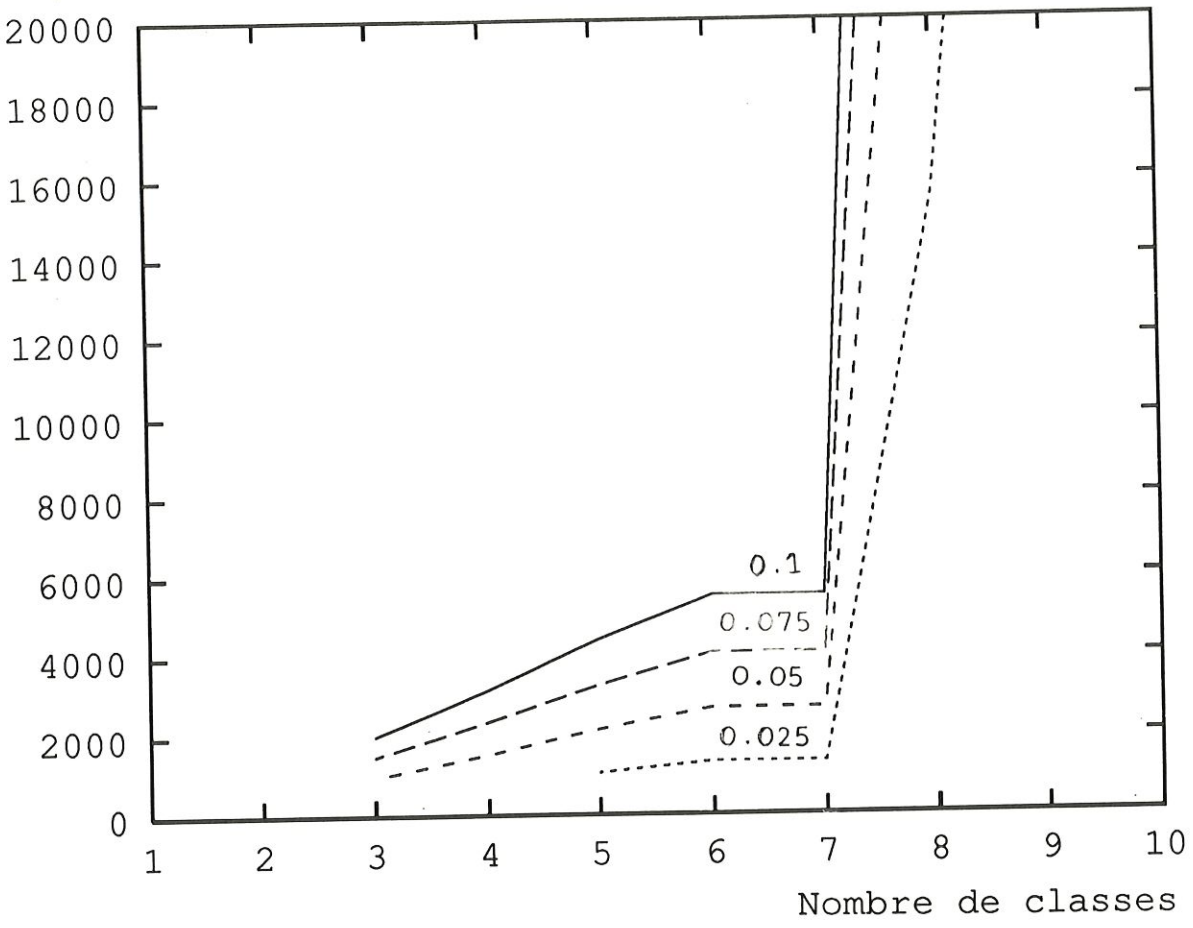


*FIG.*4.16

Les courbes K_α pour différents α ont la même allure : elles sont égales à un facteur multiplicatif près. Cela s'explique par le fait que pour un même exemple, le VOL_{tot} et les points des classes à recoller sont identiques à chaque itération. Il n'y a donc que le α qui change d'une exécution à l'autre. Le rapport d'une courbe à l'autre vaut :

$$\lambda = \frac{-\ln(1-\alpha_1)}{K_{\alpha_1}} = \frac{-\ln(1-\alpha_1)}{K_{\alpha_1}}$$

ce qui est équivalent à :

$$K_{\alpha_1} = \frac{\ln(1-\alpha_1)}{\ln(1-\alpha_2)} K_{\alpha_2}$$

Pour obtenir le nombre de classes de l'algorithme, il suffit de prendre la partie entière de l'intersection de la courbe K_α avec celle de l'écart (FIG 4.17) puisque rappelons le, la phase de recollement est interrompue si l'écart est strictement plus grand que K_α.

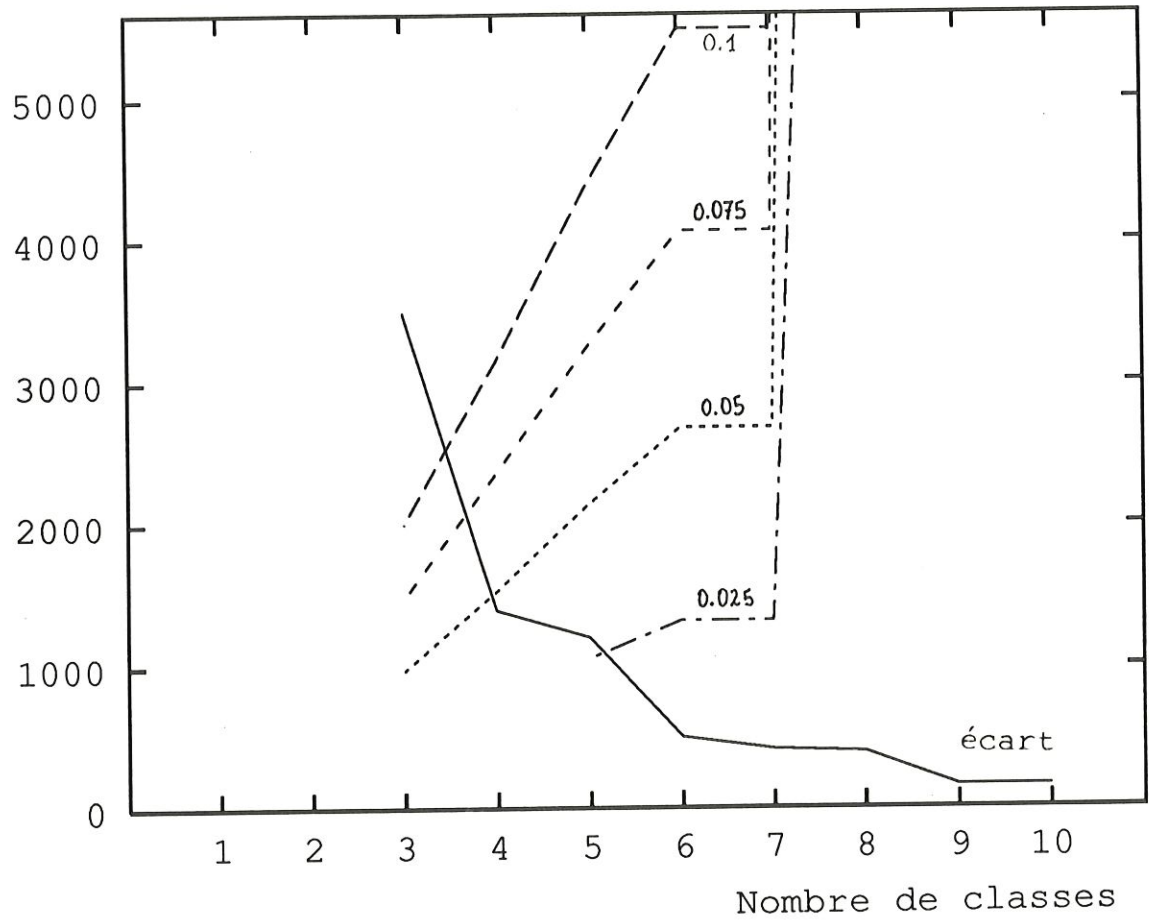


FIG.4.17

Ce graphe est aussi l'interprétation géométrique des plages plus ou moins larges de α. En effet les écarts pour aller de six à cinq classes ou pour aller de trois à deux sont significativement

importants. Et comme toutes les courbes K_α sont proportionnelles, il est normal qu'elles croisent plus souvent la courbe des écarts dans ces zones.

Si nous omettons, pour suivre, les deux grandes classes externes et que nous ne gardons que les trois classes du centre (FIG. 4.18).

*FIG.*4.18

Nous obtenons les résultats suivants:

α	*classes*
0.1	2
0.075	2
0.05	3
0.025	3

Le coude (FIG 4.19) indique une partition en trois classes. Malgré tout, le critère n'est pas stabilisé et décroit encore, de manière constante, après quatre classes.

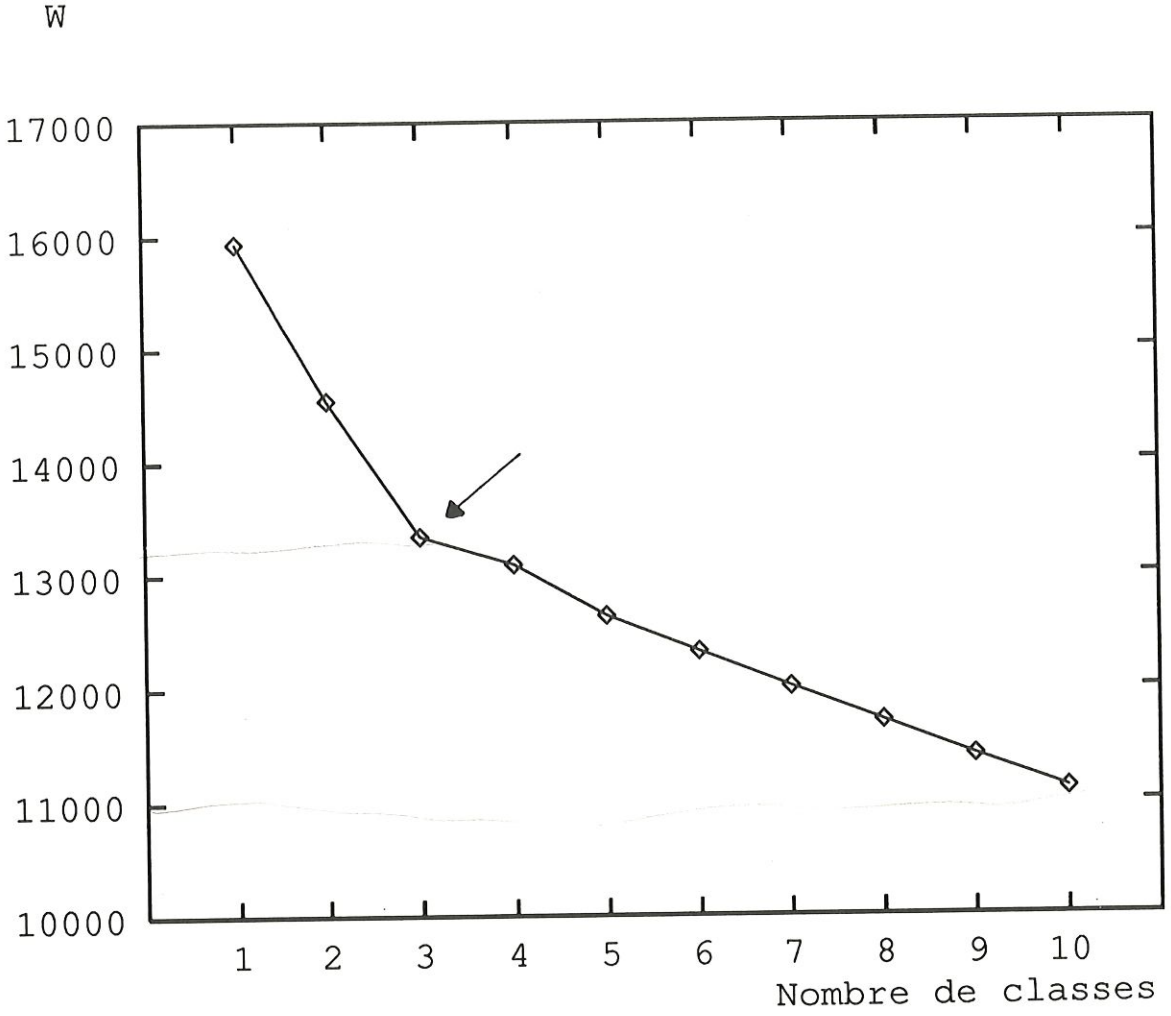


*FIG.*4.19

D'après ces résultats, nous comprenons que lorsque les cinq classes sont présentes, il est difficile de les retrouver car les trois classes centrales sont elles-mêmes difficiles à repérer. C'est pourquoi, nous allons transformer légèrement l'exemple des cinq rectangles de sorte que l'écart entre toutes

les classes augmente. (FIG 4.20)

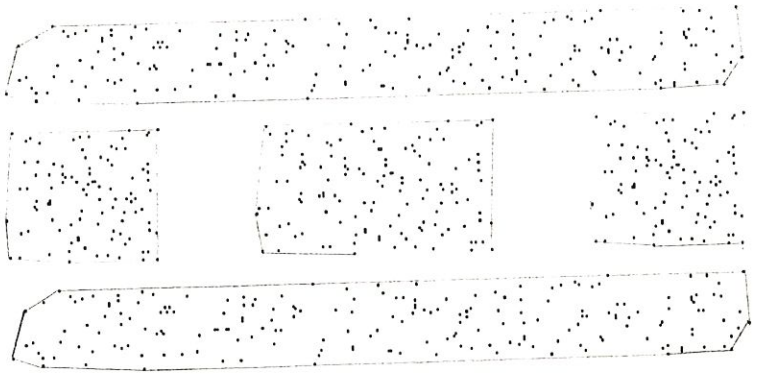


FIG.4.20

Le coude cette fois est beaucoup plus net et indique cinq classes pour la partition optimale.
(FIG 4.21)

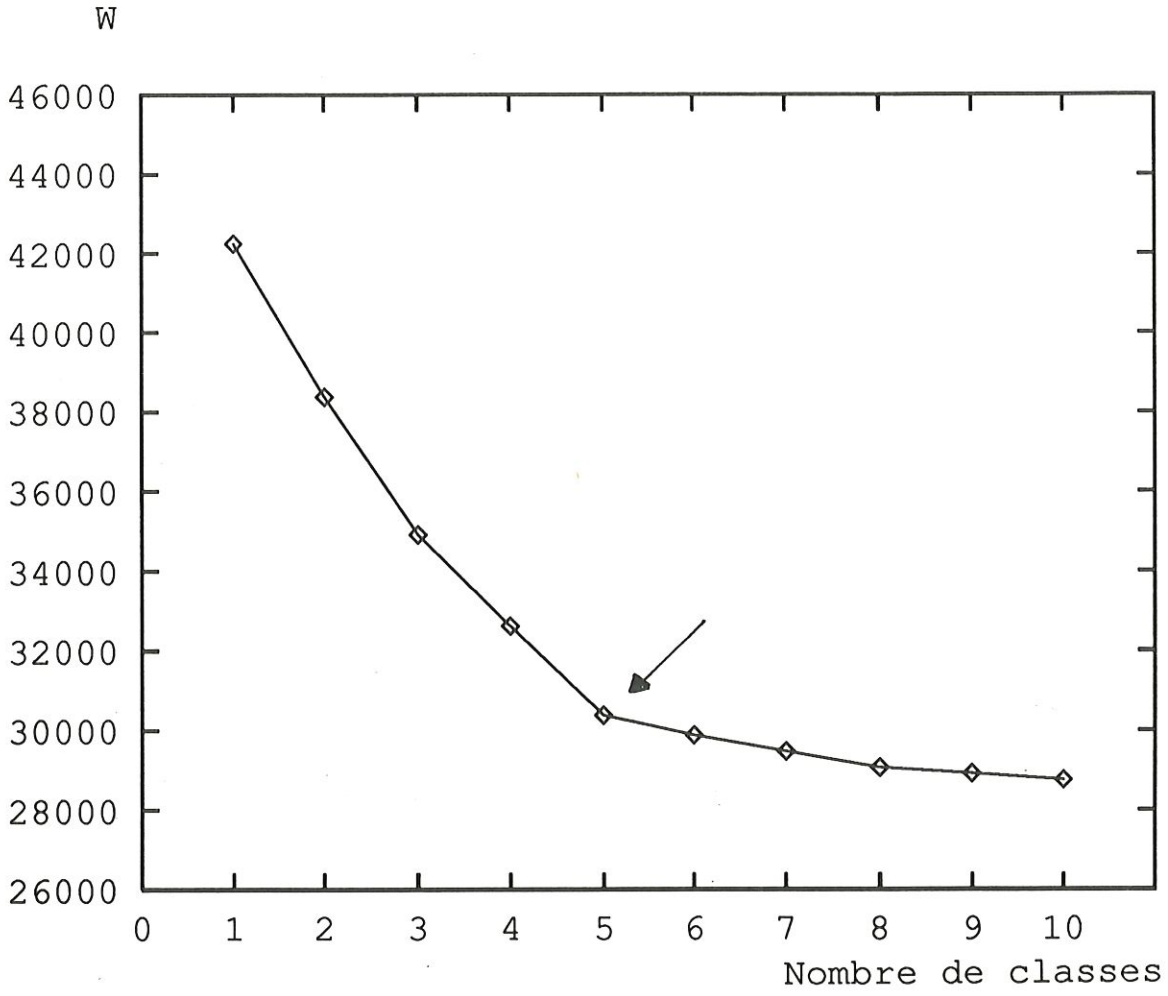


*FIG.*4.21

Mais lorsque nous comparons le tableau ci-dessous avec le précédent, nous constatons que la relation entre α et le nombre de classes n'a, apparamment pas, évolué.

α	*classes*
0.2	2
0.1	3
0.075	3
0.05	4
0.025	5

Mais ce n'est qu'une apparence. Car si nous étudions les autres aspects, voici ce que nous obtenons.

Examinons le graphe qui relie la famille K_α et l'écart, nous nous attendons à ce que la plage de α pour cinq classes soit plus grande puisque l'écart pour passer d'une configuration en six classes à une autre en cinq classes, est très grand. (FIG. 4.22)

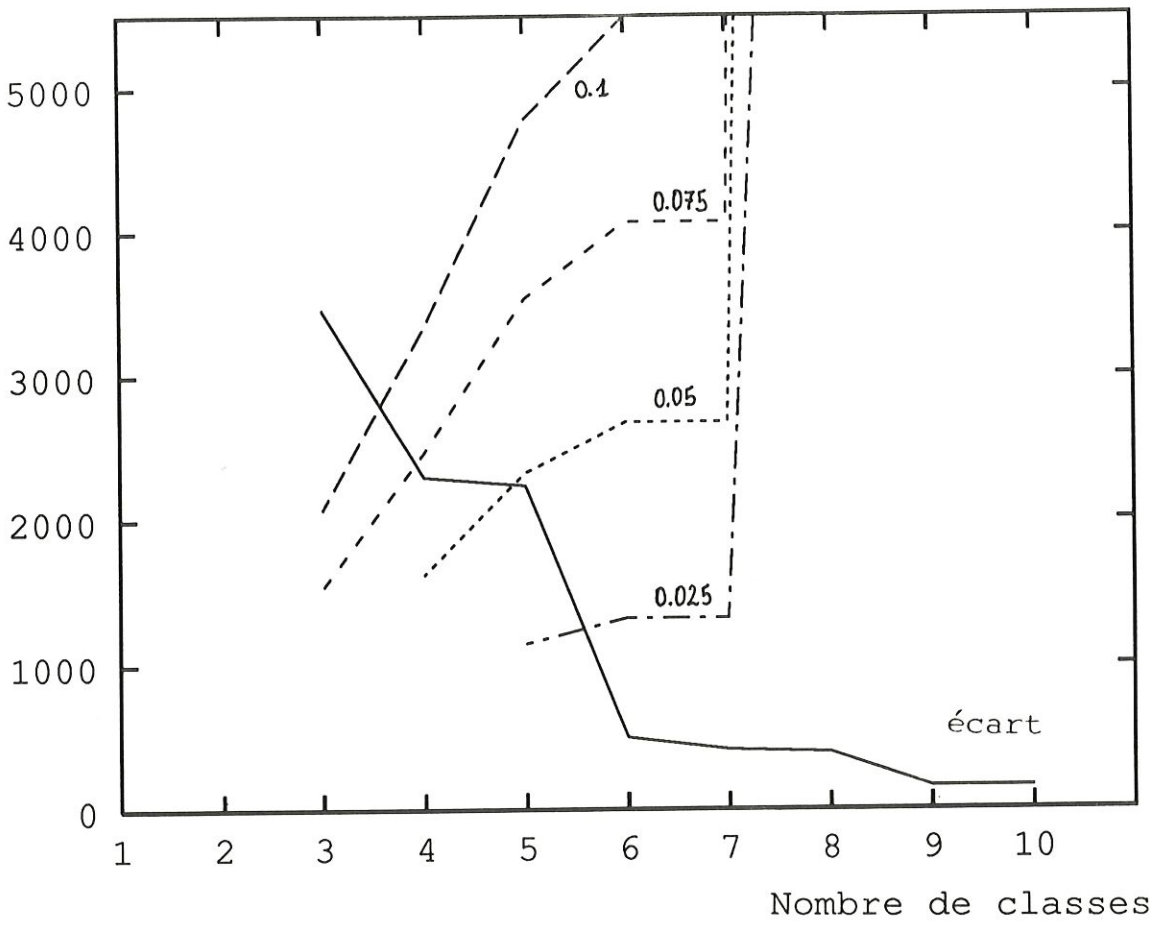
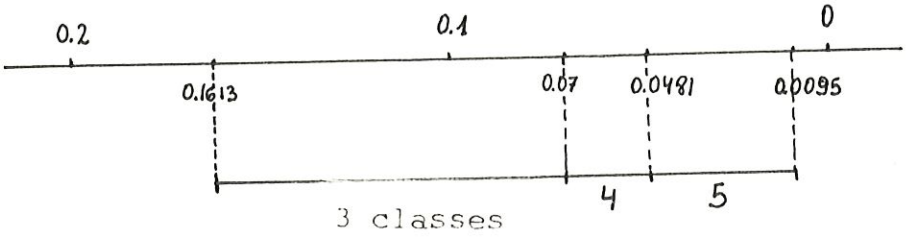


FIG.4.22

Vérifions le numériquement pour les plages de 3,4 et 5 classes.



La plage de trois classes s'est réduite en faveur de celle en cinq classes. La plage de quatre classes quant à elle, n'a pas évoluée.

Conclusion :

Nous pouvons conclure pour cet exemple que dans un premier temps, même si une structure en cinq classes apparaissait, la structure en trois classes était prépondérante. Et lorsque l'écart devient plus important, nous pouvons opter pour une partition en cinq classes.

Le test d'hypothèse et les différents graphiques le concernant, donnent les résultats que nous attendions en accord avec la méthode du coude.

Nous avons, pour terminer ce paragraphe, générer un exemple particulier : un ensemble de données sans structure ou, ce qui revient au même, une seule grande classe de 500 points. Et nous cherchons à savoir comment va réagir notre test d'hypothèse dans de pareilles circonstances.

Signalons que la méthode du coude ne donne aucune indication supplémentaire (FIG 4.23).

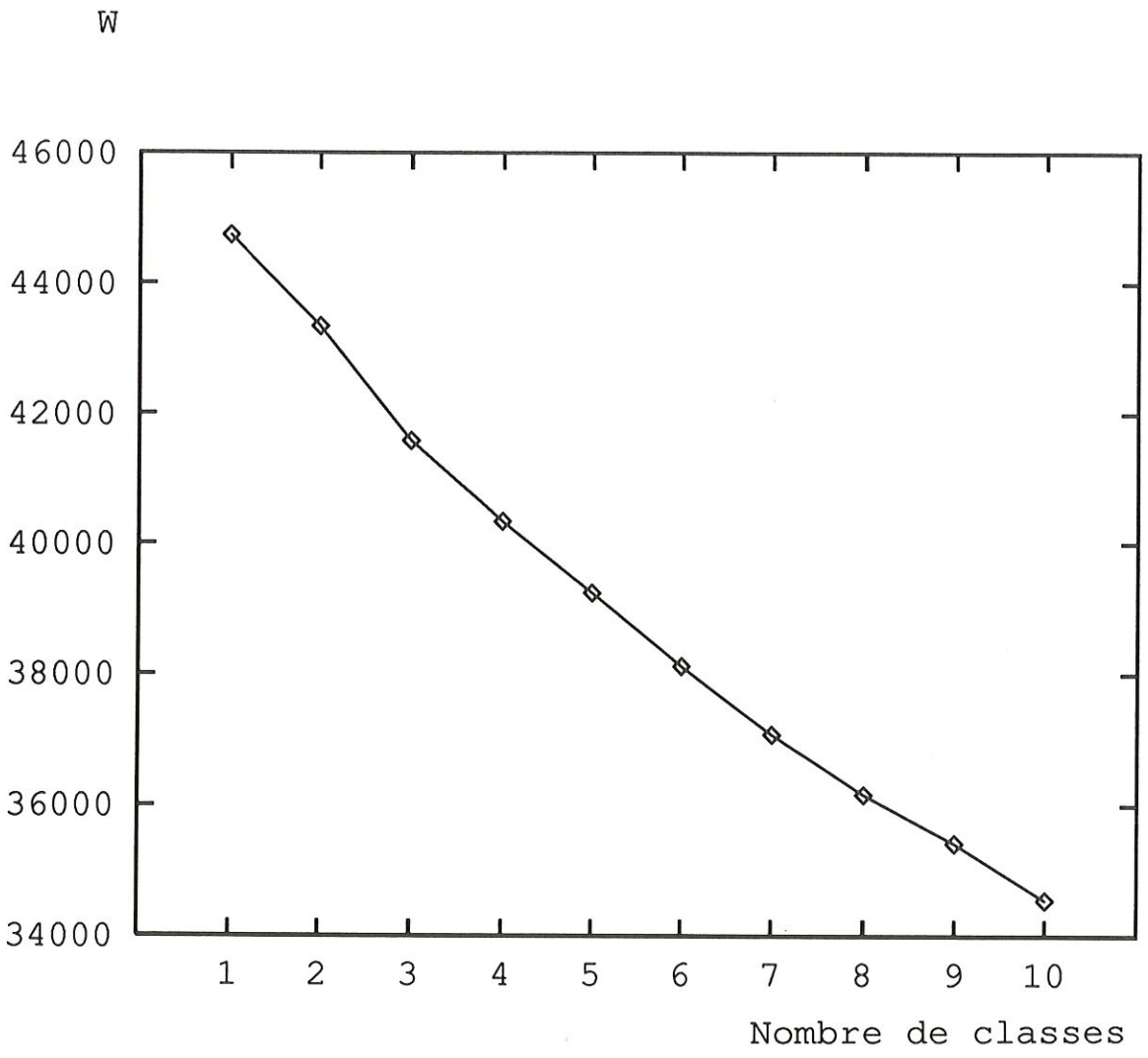


*FIG.*4.23

Voici, les résultats du recollement pour différentes valeurs de α.

α	*classes*
0.2	1
0.15	1
0.1	3
0.075	3
0.05	4
0.025	5
0.01	10

Dans l'intervalle fréquent [0.1,0.05], nous obtenons une configuration de trois (FIG 4.24) ou quatre classes (FIG 4.25).

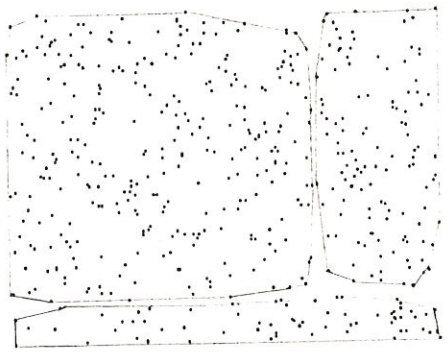


*FIG.*4.24

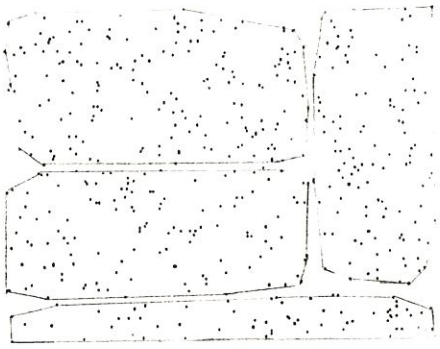


*FIG.*4.25

En examinant, comme auparavant, les valeurs de K_α et des écarts, nous obtenons :

		K_α		
classes	*ecart*	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$
10	872	8525	4025	1952
9	740	7064	3335	1623
8	917	20953	9893	4816
7	1049	9969	4707	2291
6	1115	16483	7782	3788
5	1094	9509	4490	2185
4	1240	4053	1913	931
3	1752	2881	1360	
2	1403	2472		

Pour la famille de courbes K_α, nous observons de grandes fluctuations - en dents de scie - puis une stabilisation à partir de quatre classes (FIG 4.26).

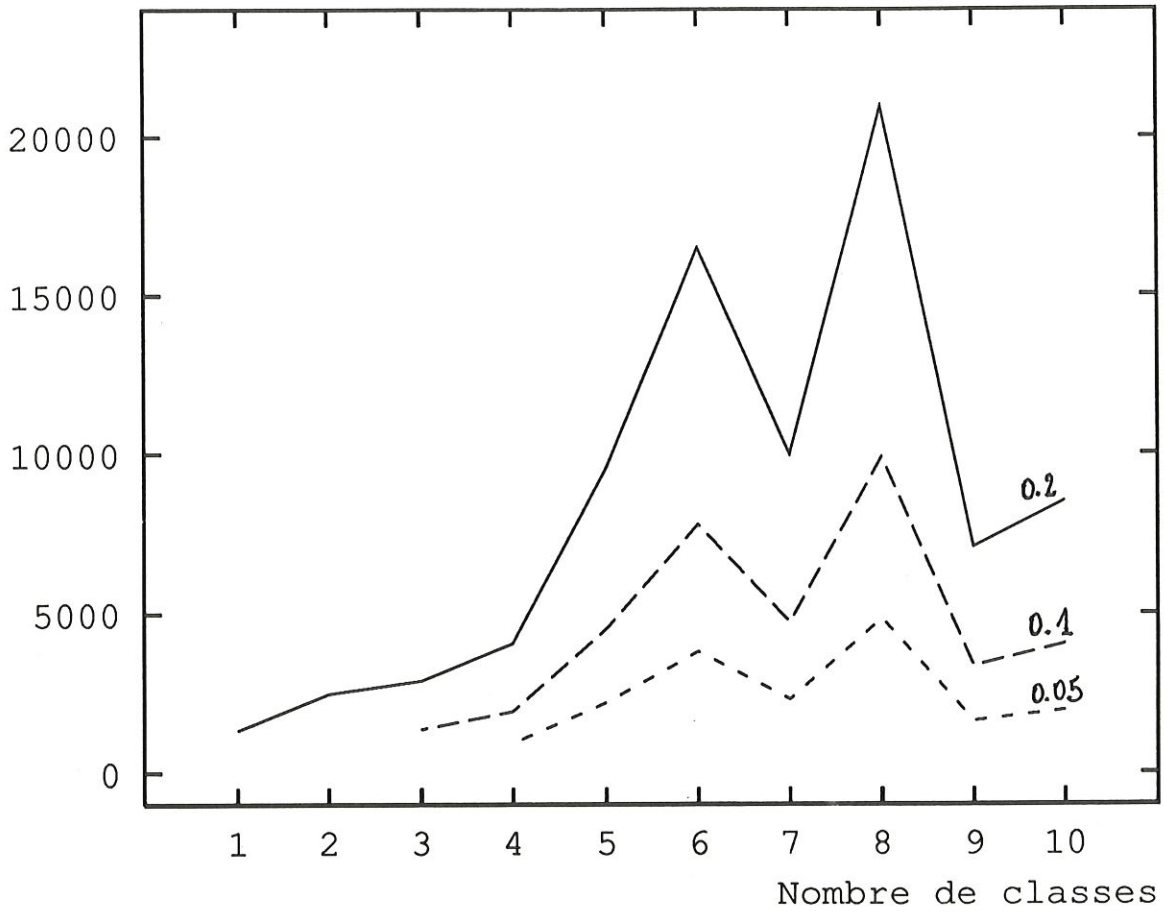


*FIG.*4.26

A l'inverse, l'écart entre les deux classes à recoller n'évolue pas pour les configurations de dix à quatre classes puis, l'écart monte de 40% de quatre à trois classes et redescend aussitôt pour

la configuration en deux classes (FIG 4.27).

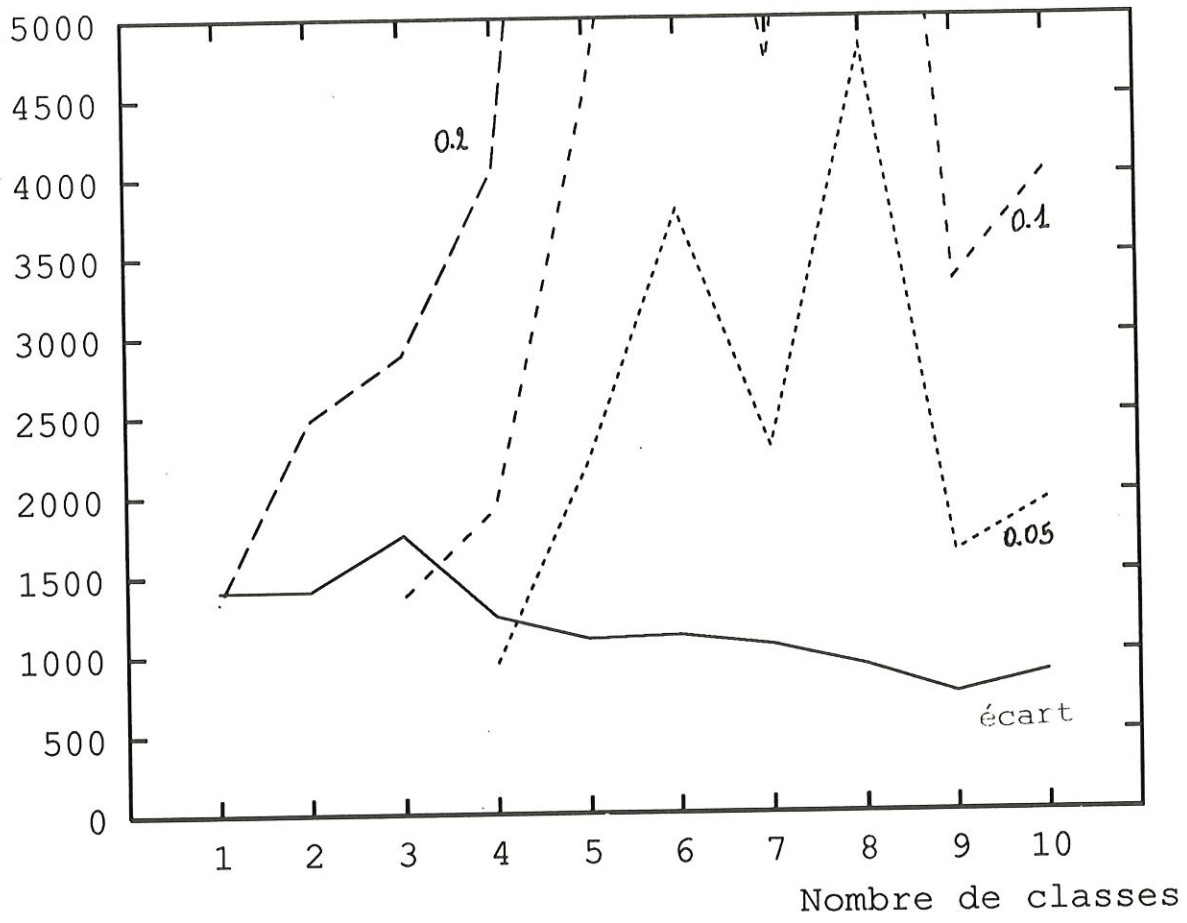
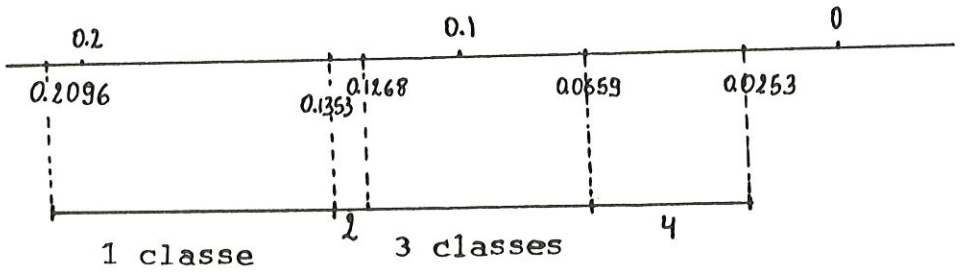


FIG.4.27

Si nous utilisons le critère des plages de α, nous obtenons des intervalles plus longs pour des répartitions en une ou trois classes.



Conclusion:

Par la méthode du coude, nous devons opter pour des données sans structure alors que la méthode par le test d'hypothèse propose une répartition des données en trois classes.

Tous les exemples que nous venons d'examiner jusqu'à présent, sont des exemples générés. Etudions à présent des données de référence traitées à de nombreuses reprises en classification automatique.

4.2 Les données de référence

4.2.1 Les données de Ruspini

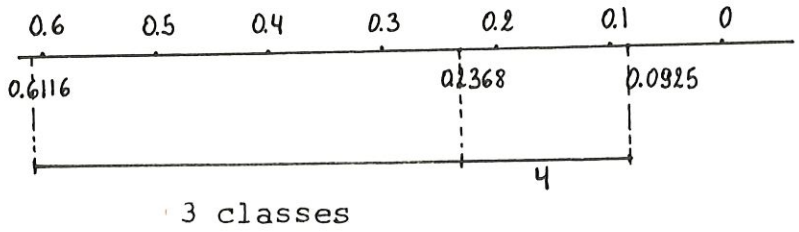
Nous avons déjà partiellement traité ces données au paragraphe 1.2.2. Pour rappel, il est très difficile de déterminer le nombre de classes de la partition optimale. nous avions considéré les partitions en trois, quatre et cinq classes. Quant à la méthode du coude, elle indiquait une partition en cinq classes.

Voyons les résultats avec le test d'hypothèse:

α	*classes*
0.3	3
0.2	4
0.1	4
0.075	5
0.05	5
0.025	6

Lorsque nous considérons l'intervalle [0.1,0.05] de α, le nombre de classes est de quatre ou cinq.

En appliquant la technique de la plage la plus longue, nous obtenons un partitionnement optimal en trois ou quatre classes.



3 classes

Conclusion :

La détermination du nombre de classes pour les données de Ruspini s'avère inextricable. Le choix entre trois, quatre ou cinq classes est difficile. Cette indécision est légitime pour quiconque regarde les données. Il est d'ailleurs difficile à l'observateur lui-même, de trouver le nombre de classes naturelles.

4.2.2 Les données de Gavaert

Ces données, en deux dimensions, de 107 points sont assez particulières. Elles sont composées de nombreuses petites classes. Nous appercevons nettement que les points appartiennent à $I\!N \times I\!N$. (FIG 4.28)

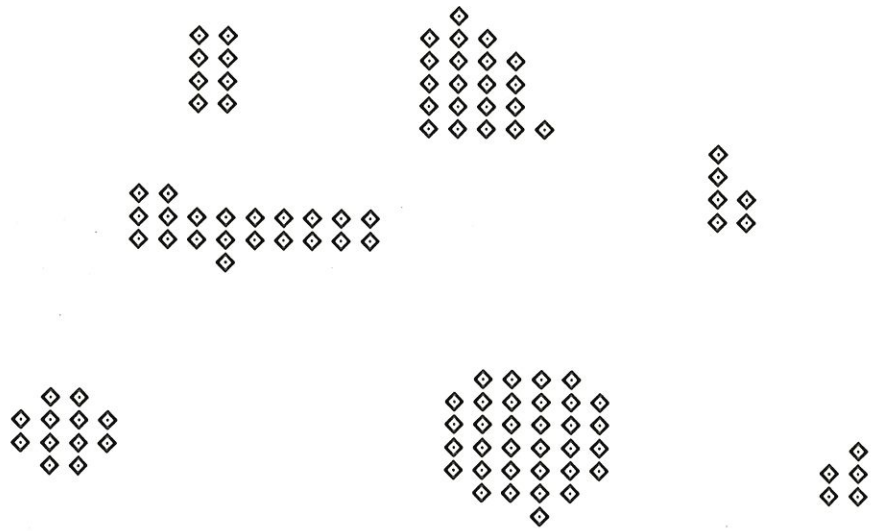


*FIG.*4.28

La courbe du critère W de classification en fonction du nombre de classes étant bien régulière, la méthode du coude ne donne aucune indication quant au nombre de classes de la partition

optimale.(FIG 4.29)

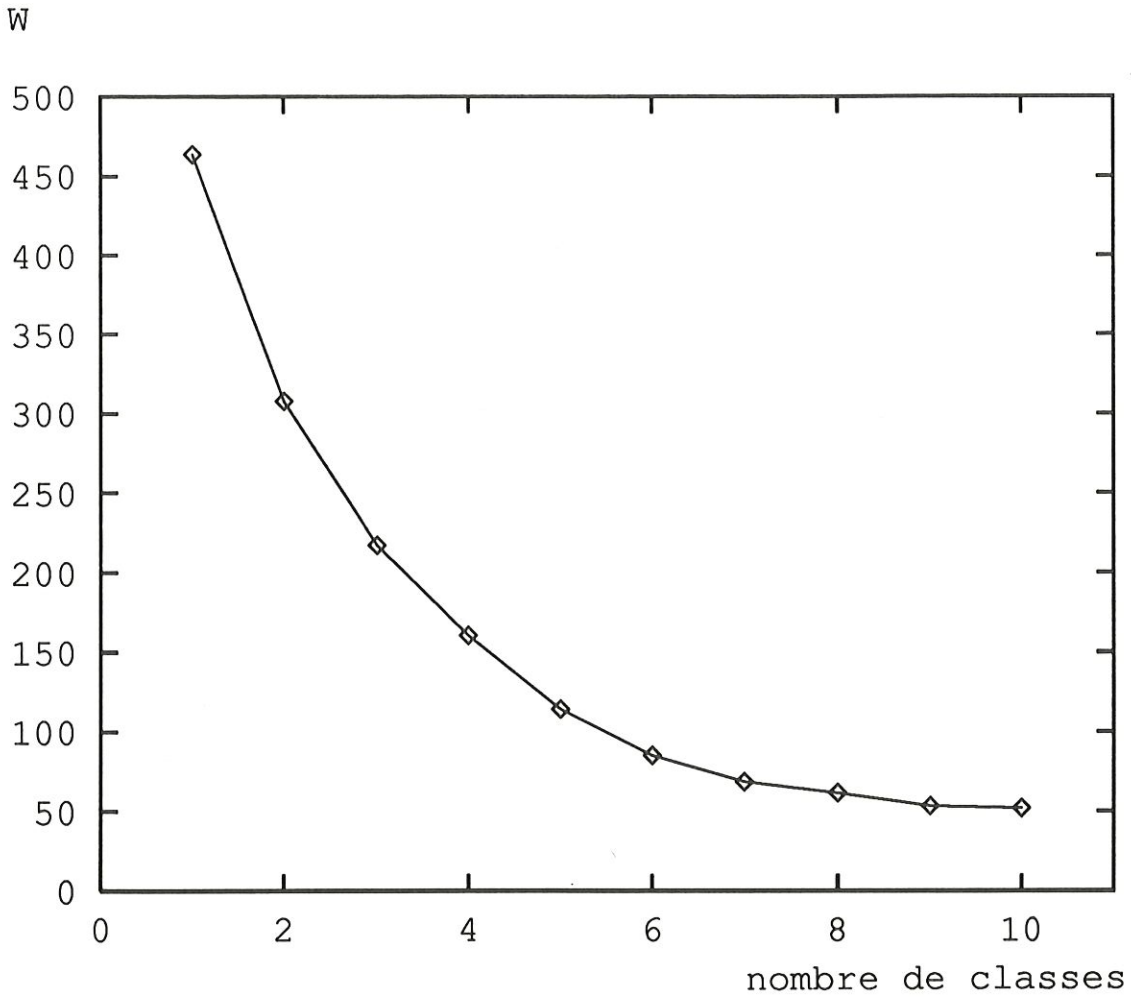
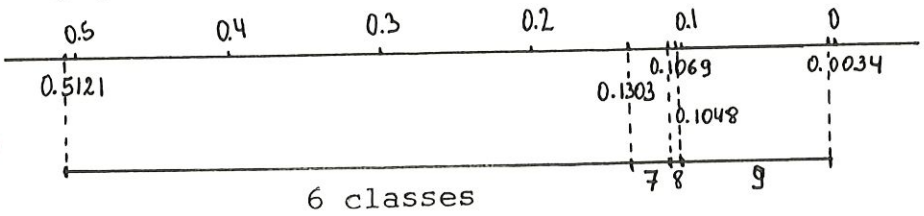


*FIG.*4.29

En revanche, le test d'hypothèse donne une multitude de configurations.

α	*classes*
0.2	6
0.15	6
0.12	7
0.1	9
0.075	9
0.05	9

Etudions les plages de α pour chaque configuration.



D'après ce schéma, nous avons le choix entre une partition en neuf ou en six classes. Et si l'on reste dans la fourchette [0.1,0.05], nous devons conclure en une partition en neuf classes.

Si nous admettons un α légèrement supérieur à 0.1, nous obtenons une configuration en six ou sept classes; ce qui est plus conforme à la réalité.

4.2.3 Les iris de Fisher

Les iris de Fisher constituent un jeu de données difficile à traiter ; il est souvent utilisé pour tester de nouvelles méthodes en classification automatique. Ce jeu de données comprend 150 points dans un espace à quatre dimensions ; chaque point représentant une fleur provenant d'une des trois variétés d'iris : Setosa, Versicolor et Virginica. Les quatres caractéristiques mesurées sur ces fleurs sont la longueur et la largeur des pétales et des sépales. Chaque variété est représentée par 50 fleurs. Le groupe d'iris Setosa est très bien séparé des deux autres. Par contre, les iris Virginica et Versicolor sont mélangés.

La méthode du coude (FIG 4.30) ainsi que le test d'hypothèse donnent les partitionnements en deux ou en trois classes :

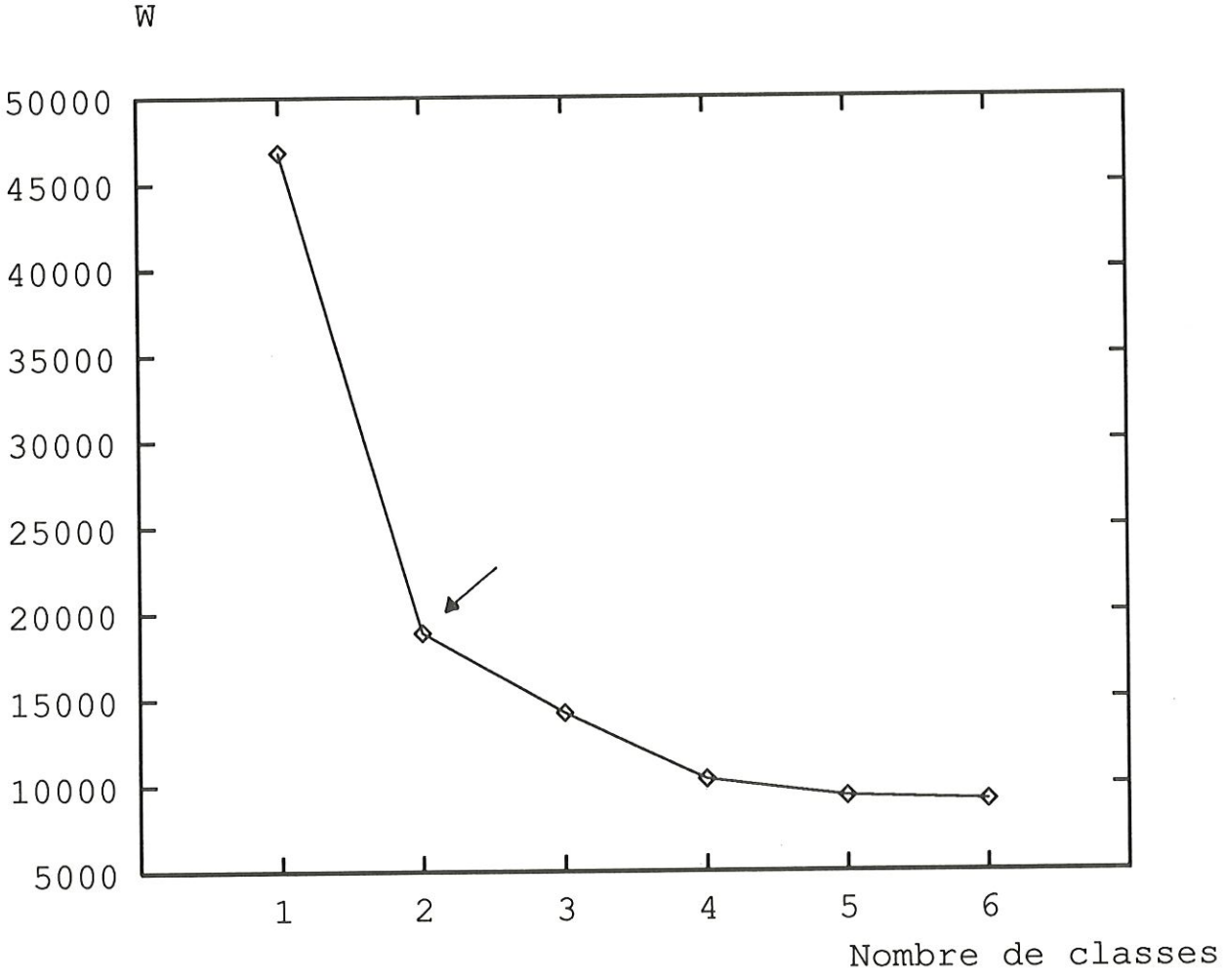


*FIG.*4.30

La partition en deux classes requiert un temps C.P.U. de 54,53 secondes contre 50,88 sec pour la partition en trois classes.

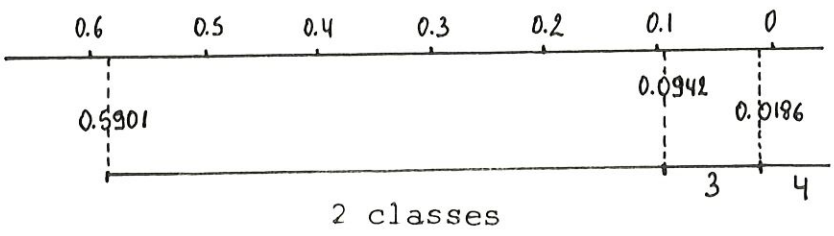
α	*classes*
0.2	2
0.1	2
0.075	3
0.05	3
0.025	3

Mais, il ne faut pas se leurrer. Si la partition en deux classes sépare les 50 Setosa des 100 autres iris, la partition en trois ne donne pas trois classes de 50 fleurs chacune. Pour des raisons déjà évoquées précédemment, la classification est celle que V. Gendarme a obtenue [5]. C'est-à-dire :

	Setosa	*Versicolor*	*Virginica*
*Classe*1	50	0	0
*Classe*2	0	15	8
*Classe*3	0	35	42

Il y a donc 29% d'iris mal affectés. Rappelons que ce résultat est dû au fait que les enveloppes convexes des classes Versicolor et Virginica ne sont pas disjointes.

Sur le graphique suivant (FIG 4.31) , l'écart pour le passage de 3 à 2 classes est très élevé, ce qui nous laisse penser que la plage de α pour deux classes doit être longue. Vérifions-le.



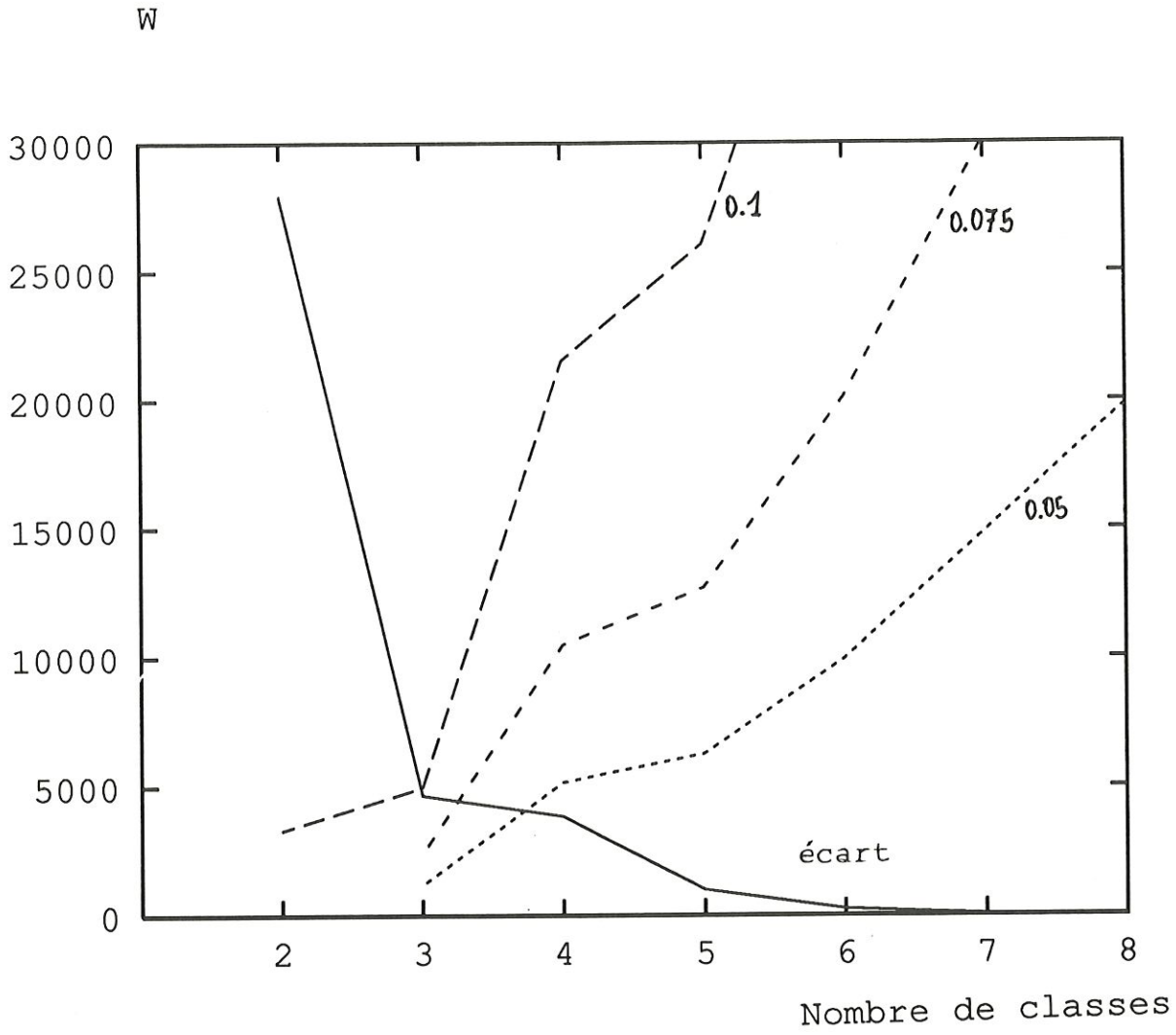


*FIG.*4.31

Au vu de tous ces résultats, nous proposons une structure de deux classes pour ces iris.

Plusieurs auteurs prétendent qu'il est arbitraire de vouloir absolument séparer ces données en trois classes. Les résultats acquis par la méthode du test d'hypothèse, conforte donc cette opinion.

4.3 D'autres données

4.3.1 Les données d'Emilromania

Restons dans les données en quatre dimensions.

Ces données portent sur la région de Vans en Ardèche. Cette région agricole est partegée en 14 types de cutures (blé, vigne, melon, courgette, ...).
Le problème est donc de déterminer l'affectation du sol dans une région à partir de données satellite.

Le coude, très marqué, renseigne une partition en trois classes comme partition optimale de ces 1830 points.(FIG 4.32)

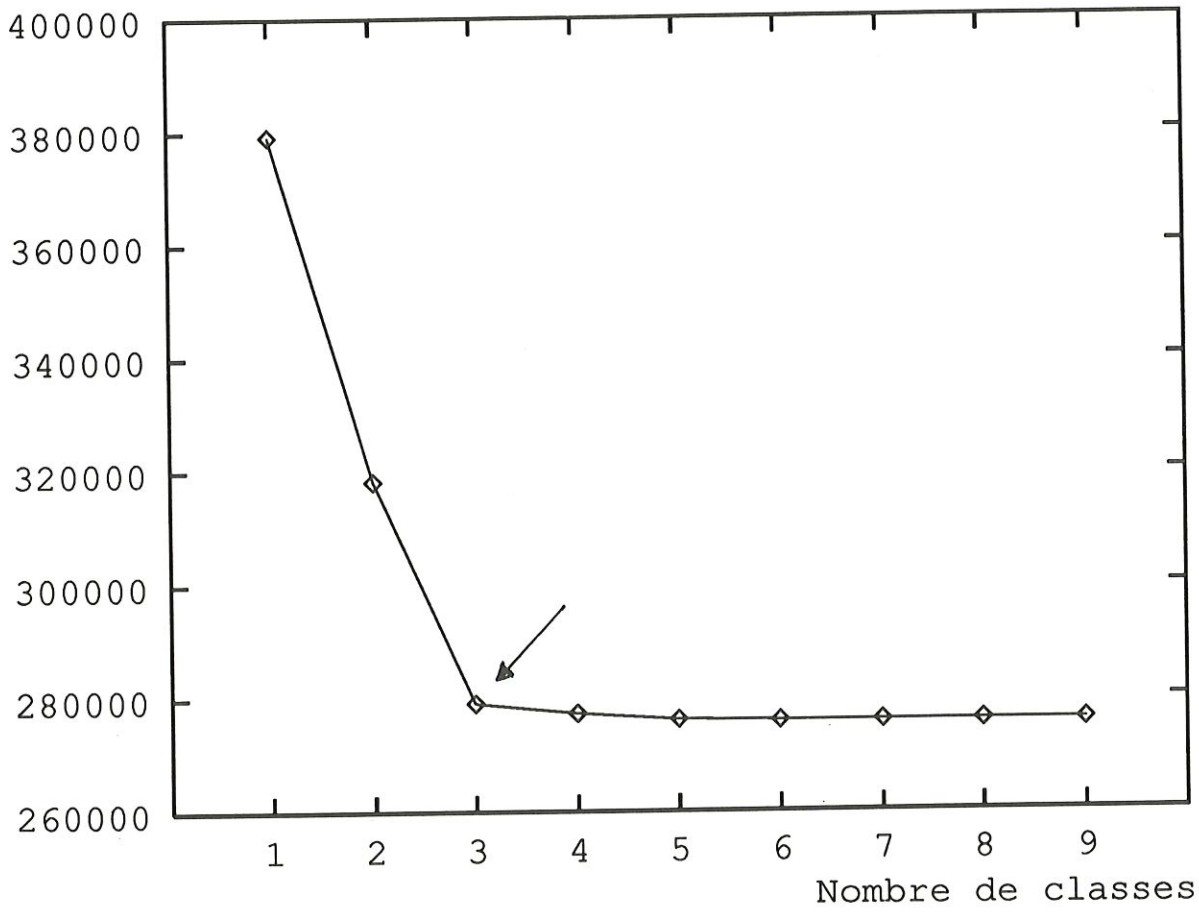


*FIG.*4.32

En revanche, le test d'hypothèse propose pour des valeurs usuelles de α, une configuration en deux classes. Il y a donc discordance entre ces deux résultats.

α	*classes*
0.1	2
0.075	2
0.05	2
0.025	2

La configuration en deux classes a été obtenue après 9 minutes et 36,95 secondes.

Mais il y a une remarque importante à formuler :

Si nous observons attentivement le tableau de reśultats suivant :

		K_α		
classes	*ecart*	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.025$
10	0	2630991	1280861	632219
9	0	1973243	960646	474164
8	23	717543	349325	172423
7	38	789297	384258	189665
6	194	415419	202241	99824
5	1073	272171	132502	65401
4	1842	563783	274470	135475
3	39104	183557	89362	44108
2	60933	4313	2099	1036

Pour $\alpha = 0.1$, K_α passe de 183557 à 4313 lorsque la configuration passe de trois à deux classes. Quarante fois moins !

Ceci s'explique par le nombre élevé de points contenus dans les deux classes à recoller.

	n_1	n_2
$8classes$	7	4
$7classes$	4	6
$6classes$	8	11
$5classes$	19	10
$4classes$	3	11
$3classes$	29	14
$2classes$	43	1787

Lors de l'évaluation du recollement des deux dernières classes en une seule, le nombre de points à recoller est donc de $n_1 + n_2 = 1787 + 43 = 1830$. Or

$$K_\alpha = \frac{-\ln(1-\alpha)}{\lambda} \text{ où } \lambda = \frac{n_1 + n_2}{VOL_{tot}} = \frac{NP_{tot}}{VOL_{tot}}$$

Donc quand $n_1 + n_2$ augmente,λ aussi. Et par conséquent, K_α diminue proportionnellement.

D'autre part, l'écart fait un bond spectaculaire dans la direction opposée pour le passage de quatre à trois classes puis encore de trois à deux classes. Sur le graphique, c'est très net.(FIG 4.33)

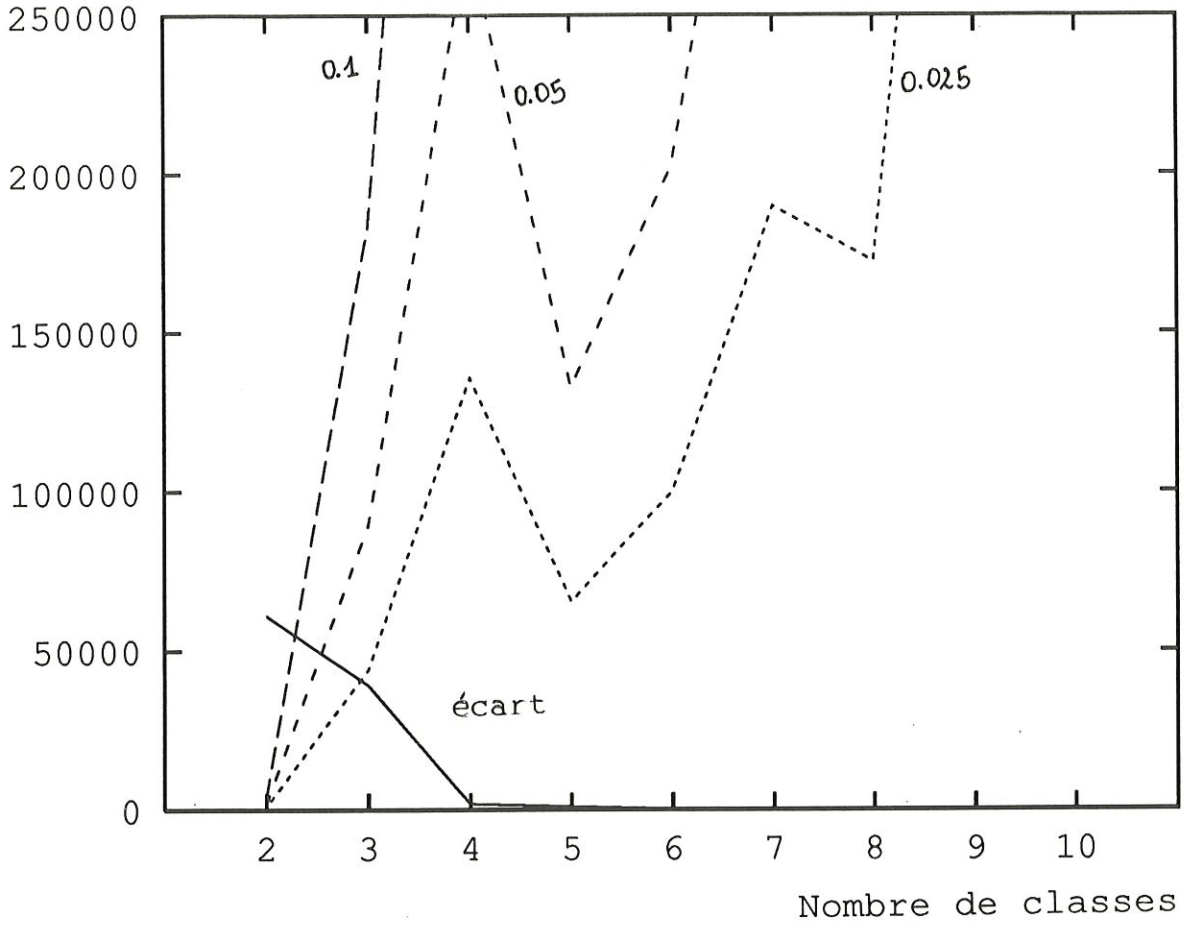


*FIG.*4.33

4.4 Conclusion

Récapitulons ce que le test d'hypothèse apporte à la détermination du nombre de classes.

Nous pouvons être très satisfaits de cette approche. Dans un intervalle raisonnable de α et pour un ensemble de données composé de classes naturelles suffisamment marquées, le test d'hypothèse arrête le recollement à la partition optimale. Sans compter que ces résultats sont souvent conformes à la méthode du coude.

Cependant, il arrive parfois que le test d'hypothèse ne donne pas le même résultat que le coude. Dès lors, il faut envisager comme possible, les deux cas de figure et pourquoi pas, trancher en faveur de l'une d'entre elles par des considérations autres que mathématiques.

Nous avons vu également que lorsque la méthode du coude reste sans réponse, le test d'hypothèse peut trouver une configuration dans les données.

Nous devons donc, considérer l'approche par les tests d'hypothèse et la méthode du coude comme deux sources d'informations à confronter et essayer d'en tirer le meilleur parti.

Enfin, les temps C.P.U. calculés sont très semblables à ceux de V. Gendarme.

Bien sûr, tout n'a pas été résolu et quelques domaines restent incertains : l'exemple de deux classes reliées par un pont n'a pas été traité. Il fera l'objet d'une étude approfondie dans le prochain chapitre.

Nous pourrions aussi, appliquer un test similaire pour la partie "division" de l'algorithme et ainsi, ne plus devoir fixer une valeur arbitraire à v.

Ou encore, changer d'approche comme le propose G Celeux et E Diday [2] , en se basant sur l'optimisation pour résoudre le problème du choix du nombre de classes : elle consiste à ajouter au critère à minimiser le coût de création d'une classe. Mais en pratique, ce coût peut être difficile à déterminer.

Enfin, pour affiner la méthode, nous pourrions appliquer le test d'hypothèse sur un algorithme dont la phase de recollement est plus appropriée.

Chapitre 5

Nouvel algorithme de classification

5.1 Introduction

Nous avons vu dans les chapitres précédents que le test permet de déterminer le nombre k de classes de la partition optimale. L'algorithme que nous avons utilisé est basé sur les hypervolumes et la maximisation du vide. Malgré ses performances, il ne résout pas tous les problèmes.

Voici d'ailleurs les points sensibles :

- lorsqu'il y a un pont entre les classes, la classification est erronnée ;
- nous fixons le nombre v de classes pour l'étape de division ;
- nous utilisons un hyperrectangle contenant tous les points à classer pour faciliter la division c'est à dire l'obtention des v classes.

Illustrons ce premier point :

Voici l'exemple de deux classes carrées reliées par un pont (FIG. 5.1)

*FIG.*5.1

et la classification obtenue par l'algorithme décrit au chapitre 1. (FIG. 5.2)

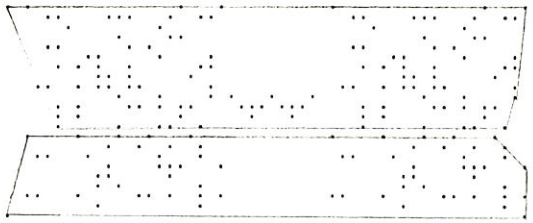


*FIG.*5.2

Cette répartition en deux classes est loin d'être idéale. Il vaudrait beaucoup mieux que la séparation entre les deux classes soit verticale et qu'elle se situe juste au milieu du pont.

Pour cette classification, la méthode du coude indique qu'il n'y a pas de structure dans les données (FIG 5.3) et dès lors, nous devons considérer la classification en une seule classe (FIG

5.1)

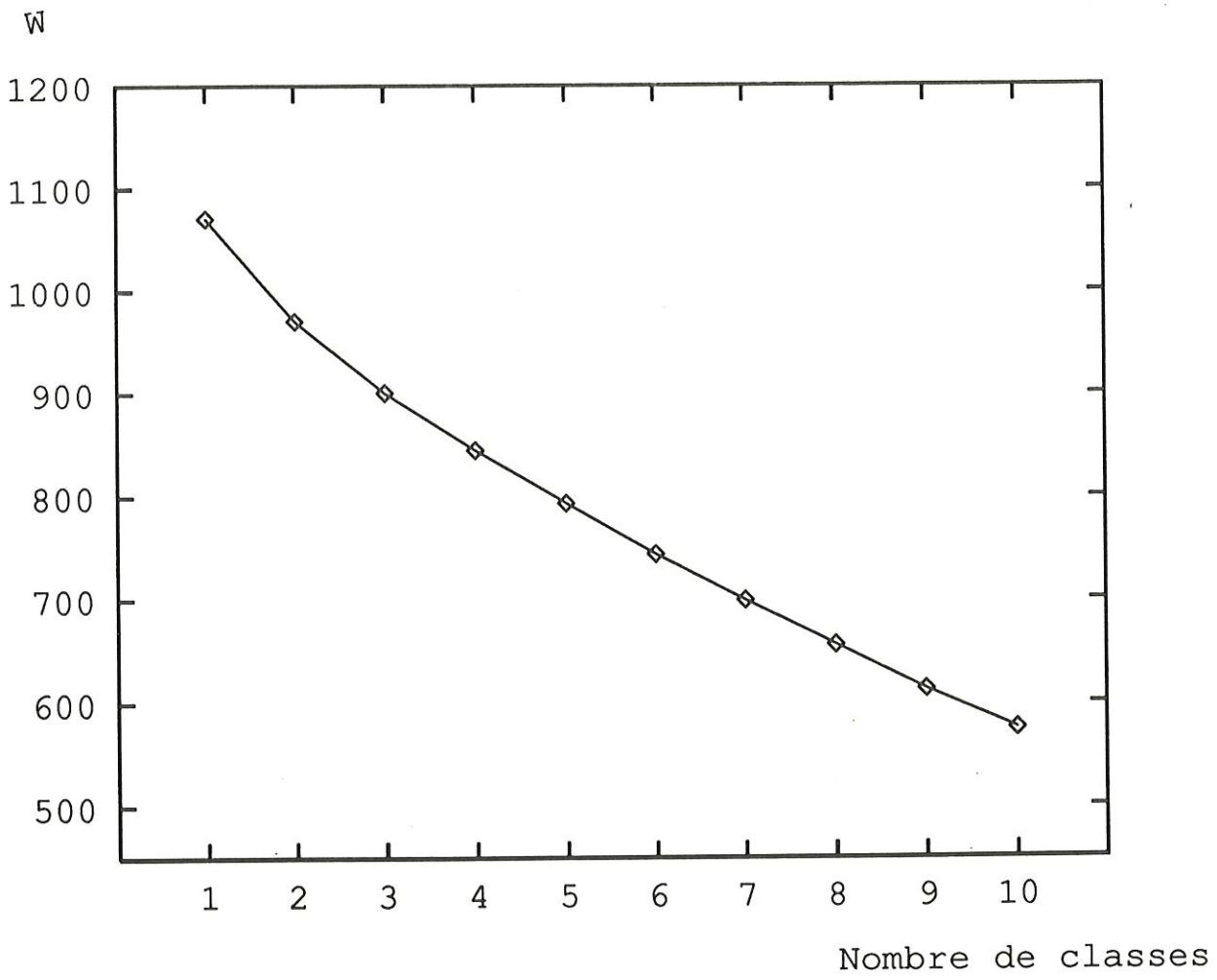


*FIG.*5.3

Le test d'hypothèse propose une partition en deux classes. Pour des raisons déjà évoquées, cette classification est celle de la figure FIG 5.2 .

α	*classes*
0.15	2
0.1	2
0.075	2
0.05	2
0.03	3
0.025	3

C'est pour pallier ces déficiences que le Professeur RASSON a proposé la méthode qui suit.

5.2 La méthode

L'algorithme comporte trois phases :
La première décrit comment obtenir le convexe de départ, la deuxième explique comment ce convexe est divisé enfin, la dernière partie concerne le recollement.

5.2.1 La recherche du convexe de départ

La méthode précédente proposait de remplacer l'enveloppe convexe des données par un hyperrectangle contenant tous les points. Cet hyperrectangle était très utile pour la division en deux.

Or, nous savons que l'algorithme basé sur les enveloppes convexes exactes donne de très bons résultats mais demande un grand temps de calcul et beaucoup de place mémoire.

C'est pourquoi, nous allons prendre l'approximation de l'enveloppe convexe suivante .

Dans le repère canonique, nous prenons pour chaque axe, la coordonnée minimale et la coordonnée maximale. Nous obtenons ainsi $2 \times DIM$ points extrêmaux.

Ensuite, nous calculons les valeurs et vecteurs propres de la matrice carrée $A = X^tX$ de dimension $DIM \times DIM$ où X est la matrice qui contient les coordonnées des points.

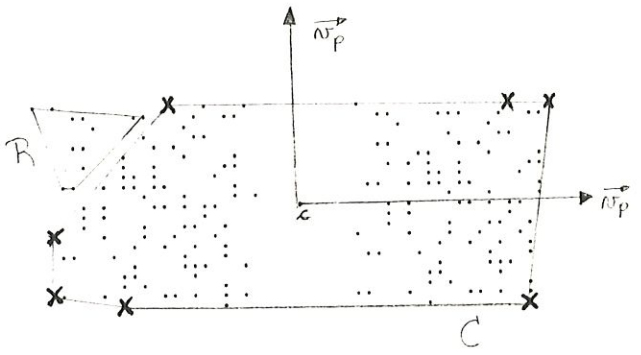
Nous avons donc les DIM axes factoriels des données. Dans ce nouveau repère, nous recherchons - toujours selon le même principe - $2 \times DIM$ points extrêmaux. Nous avons donc au maximum $4 \times DIM$ points.

Le convexe de départ sera déterminé par l'enveloppe convexe de ces points et nous le noterons C.

Il faut remarquer que :

- Il est assez fréquent d'avoir des sommets identiques.
- C ne contient pas nécessairement tous les points.
- En dimension 2, nous obtenons un convexe à huit côtés tout au plus.

Visualisons cette recherche sur un schéma.



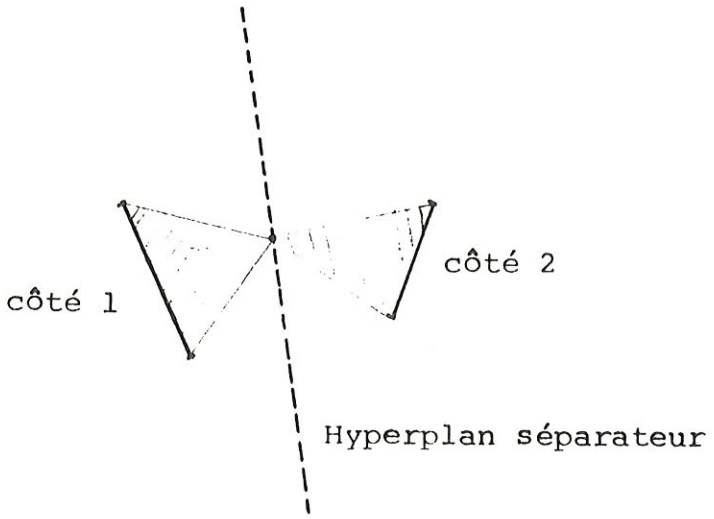
X : 7 points extrêmaux
C : convexe à 6 côtés
R : classe reste

5.2.2 Division du convexe C

Nous utilisons l'algorithme heuristique de division suivant proposé par Ph. Meunier et J.P. Rasson [10]

Chaque face du convexe C est supportée par un hyperplan. Pour chaque paire de faces non contigües de C, nous définissons un hyperplan séparateur de telle sorte que chaque point appartenant à cet hyperplan, possède la propriété suivante :

le volume de la pyramide définie par ce point et la face du premier hyperplan, ainsi que le volume de la pyramide définie par ce point et la face du second hyperplan sont égaux.



Nous obtenons de cette façon autant d'hyperplans séparateurs que de faces dans C. Nous notons ce nombre *nbh*.

Nous divisons C par ces hyperplans et nous obtenons au plus 2^{nbh} classes. Finalememt, nous avons au maximum $(2^{nbh} + 1)$ classes. Il y a en effet une classe "reste" qui contient les points hors de C. L'avantage, c'est que le nombre v de classes obtenu après division n'est pas un nombre décidé arbitrairememt. L'inconvénient, c'est qu'il peut être très élevé. Si nous prenons le pire des cas en dimension 2, nous pouvons avoir $2^{4*2} + 1 = 1057$ classes!!! Un nombre astronomique, que nous atteignons rarement dans la pratique.

5.2.3 Recollement

Cette phase est similaire à celle expliquée au paragraphe 1.2.2. Afin de déterminer le nombre de classes, nous pouvons soit appliquer la méthode du coude, soit la méthode par le test d'hypothèse, puisque le schéma algorithmique correspond aux exigences de l'implémentation du test.

5.3 Exemples

Traitons avec ce nouvel algorithme, l'exemple introductif (FIG. 5.1) des deux classes reliées par un pont. La classification que nous obtenons est très bonne (FIG 5.4)

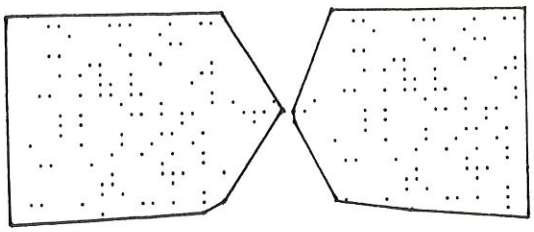


*FIG.*5.4

Dans ce cas, comme la classification des données en deux classes est bonne, il y a lieu de se demander quel est le nombre de classes de la partition optimale ?

La méthode du coude indique une partition en deux classes (FIG. 5.5)

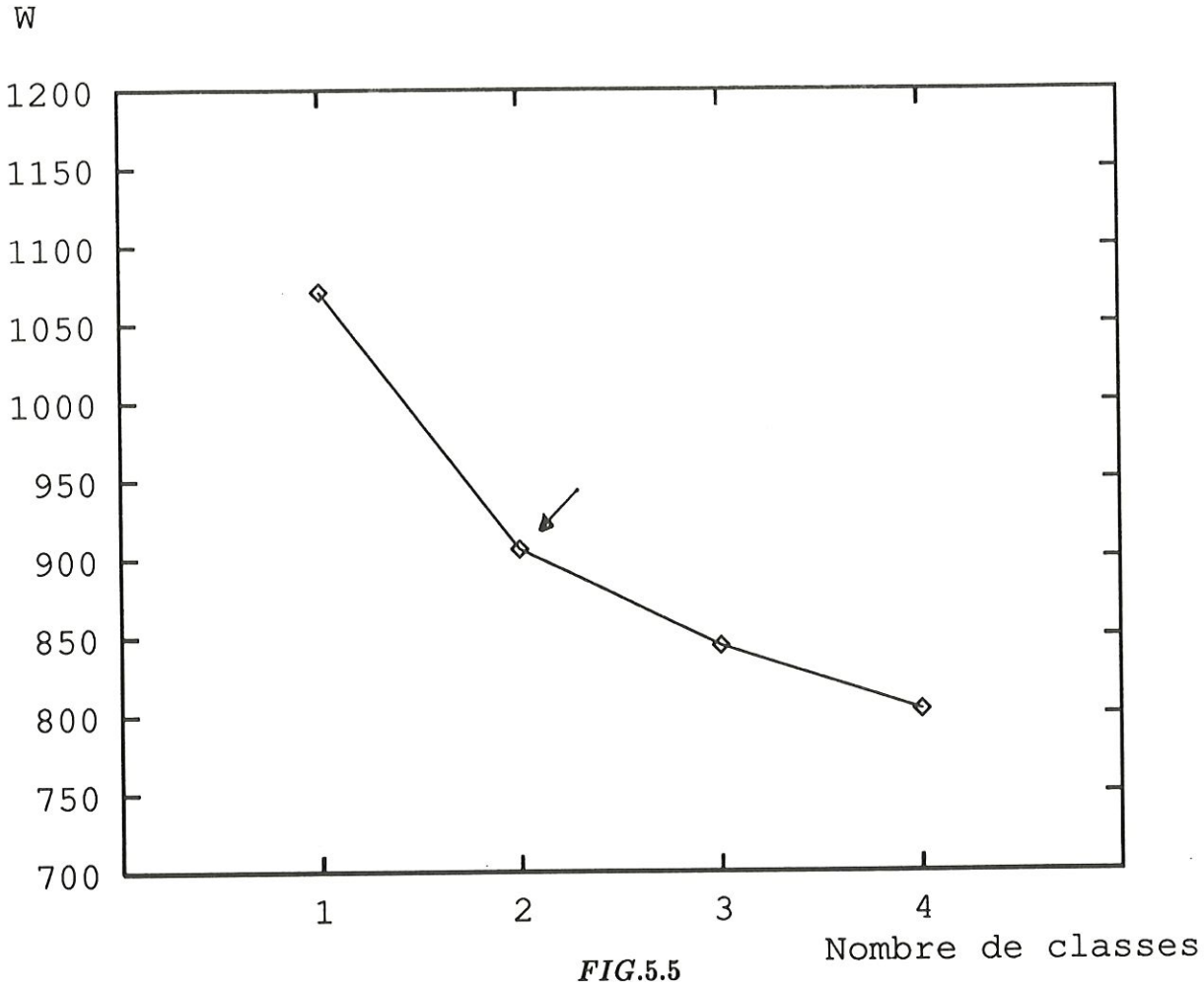


*FIG.*5.5

Cet avis est partagé par le test d'hypothèse.

α	*classes*
0.2	1
0.1	2
0.075	2
0.05	2
0.04	2
0.035	3
0.025	4

Conclusion :

Pour ce nouvel algorithme, la bonne classification en deux classes est donnée à la fois par la méthode du coude et par le test d'hypothèse.

Voici un second exemple de trois classes rectangulaires (FIG 5.6)

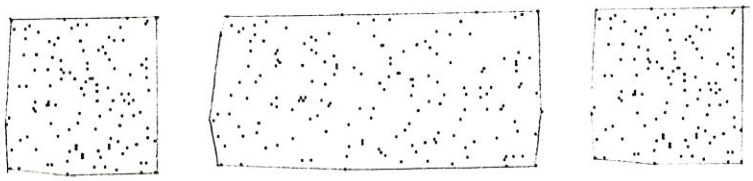


FIG.5.6

Il illustre d'une part l'existence d'une classe hors de C et d'autre part le nombre plus élevé de classes lors de la division. Nous dénombrons pas moins de 15 classes. (FIG.5.7)

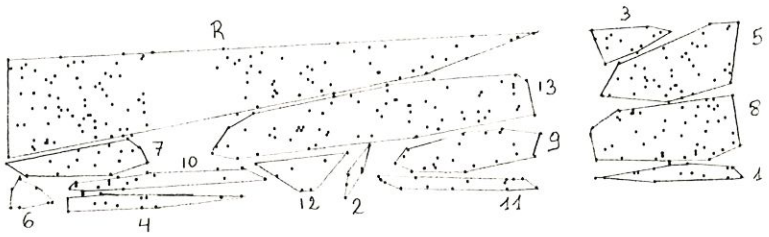


FIG.5.7

La classification obtenue n'est pas très bonne : voici la configuration des données en quatre classes (FIG 5.8).

*FIG.*5.8

Pour obtenir la configuration en trois classes, il suffit de recoller les classes 3 et 4.

Enfin, voici le partitionnement en deux classes (FIG 5.9)

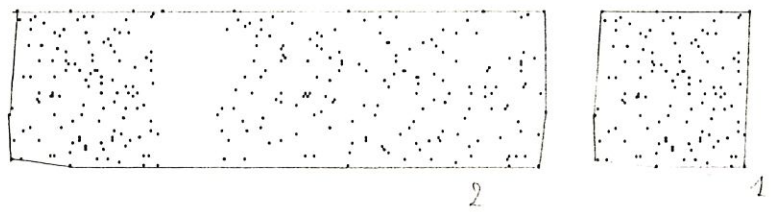


*FIG.*5.9

5.4 Conclusion

Nous le voyons, l'algorithme décrit dans ce cinquième chapitre est loin de résoudre le problème de classification. L'exemple du pont de l'introduction est une exception.

Indépendamment de ce résultat, vu le nombre non limité de classes lors de la division, la place mémoire pour stocker ces informations est très importante.

Ultime conclusion, l'algorithme basé sur les hypervolumes et la maximisation du vide reste la meilleure solution aux problèmes de classification. Si d'aventure, elle ne marche pas dans un cas ou l'autre, essayer toujours l'autre algorithme. On ne sait jamais!

Conclusion générale

Au départ de ce travail, deux problèmes nous étaient posés :

- déterminer le nombre de classes de la partition optimale.

- obtenir une bonne classification lorsqu'il y a un pont entre les classes.

Pour déterminer le nombre de classes de la partition optimale, nous avons utilisé une approche courante en statistique : le test d'hypothèse.

Combiné avec la méthode, plus traditionnelle, du coude, il permet de résoudre à la fois des exemples théoriques et pratiques.

Par contre, le problème du pont entre les classes par l'implémentation d'un nouvel algorithme est un échec. Nous avons dû abandonner cette approche et la question du pont reste donc toujours ouverte.

Appendix A

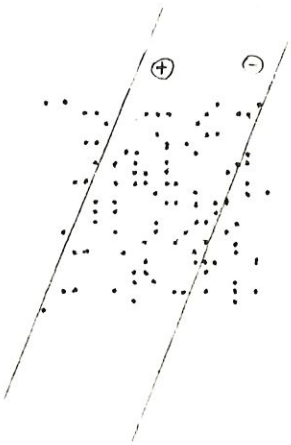
Génération de points suivant une loi uniforme

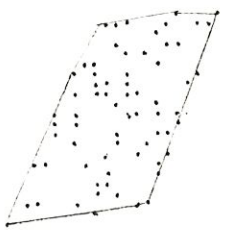
Nous présentons dans cette annexe, le listing du programme utilisé pour générer un certain nombre de points suivant une loi uniforme.

Après avoir simulé des points dans un rectangle déterminé par quatre sommets AE, BE, CE, DE, plusieurs possiblilités s'offrent à nous :

- soit nous gardons ce rectangle
- soit nous obtenons un polygône :

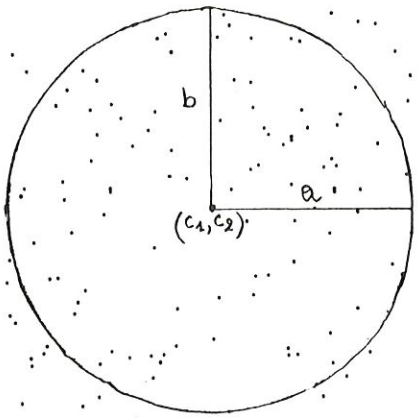
En donnant les équations de droites des côtés et le caractère positif ou négatif qui permet de définir pour chaque droite le demi-plan à conserver, nous obtenons le polygône en éliminant les points générés dans le rectangle qui sont dans l'autre demi-plan.





Enfin, l'algorithme génère également des points dans des ellipses. L'utilisateur doit fournir :

- les coordonnées du centre (C_1, C_2)
- a : le demi-axe en X
- b : le demi-axe en Y



La génération est basée sur le même principe. Nous générons les points dans le rectangle $[C_1 - a, C_1 + a] \times [C_2 - b, C_2 + b]$ puis nous retirons ceux qui ne vérifient pas l'inéquation :

$$\frac{(x - C_1)^2}{a^2} + \frac{(y - C_2)^2}{b^2} \leq 1$$

Le nombre maximal de points dans une classe est fixé à 1000, le nombre maximal d'ellipses et de polygônes à 10.

PROGRAM GENEREPLUS

PRÉSENTATION DU PROGRAMME :

Ce programme génère aléatoirement N points dans :
- un rectangle

et/ou
- un polygône

et/ou
- une ellipse

Il fait appel à une fonction de IMSL : RNUNF qui génère un nombre uniforme sur [0,1].

TABLE DES CONSTANTES :

NBRMAX : Nombre maximal de polygônes ou d'ellipses a générer.

PCINMAX : Nombre maximum de points dans chaque figure.

TABLE DES VARIABLES DU PROGRAMME :

AE,BE,CE,DE : Variables réelles représentant les sommets du rectangle dans lequel on génere les points.

I,J,K,L : Variables entières de boucle.

N : Variable entière contenant le nombre de points que l'on génère.

R : Tableau d'entiers de dimension 100 sur 2 dans lequel sont rangées les parties entières des coordonnées des points générés.

X : Tableau de réels de dimension 100 sur 2 dans lequel sont rangées les coordonnées des points générés.

Y,Z : Vecteurs de réels de dimension 100. Ce sont des vecteurs intermédiaires utilisés par la génération de nombres uniformes sur [0,1].

NBR : Nombre de rectangles et de polygônes donné par l'utilisateur.

NBE : Nombre d'ellipses donné par l'utilisateur.

C1,C2 : Coordonnées du centre de l'ellipse donné par l'utilisateur.

NDR : Nombre de droites pour transformer le rectangle en polygône.

G : Variable entière qui compte le nombre de points effectif dans un polygône ou une ellipse.

POSI : Vecteur qui contient le caractère positif (1) ou negatif (2) du demi-plan a considérer.

A,B : Le demi-axe en X et en Y donné par l'utilisateur.

COEFA,COEFB,COEFC : Vecteur qui contient les coefficients a,b,c pour déterminer la droite : ax+by+c=0.

```
C
C     DÉCLARATION DES CONSTANTES ET DES VARIABLES :
C     ---------------------------------------------

      INTEGER NBRMAX,POINMAX
      PARAMETER(NBRMAX10,POINMAX=1000)


      INTEGER I,J,K,N,ISEED,R(POINMAX,2),NBR,NBE,C1,C2,NDR,G,POSI(10),L

      REAL AE,BE,CE,DE
      REAL Y(POINMAX),Z(POINMAX),X(POINMAX,2),A,B,COEFA(10),COEFB(10)
      REAL COEFC(10)
      DOUBLE PRECISION EQUA

      CHARACTER*20 NOMF

      LOGICAL GARDE

      INTEGER INT
      INTRINSIC INT

      REAL RNUNF,RNOPT,RNSET
      EXTERNAL  RNUNF,RNOPT,RNSET
C
```

LE PROGRAMME :

```
      WRITE(*,*)'QUEL EST LE NOM DU FICHIER :'
      READ(*,21)NOMF
21    FORMAT(A20)

      OPEN(UNIT=7,FILE=NOMF,STATUS='NEW')

      WRITE(*,*)'QUEL EST LE NOMBRE DE POLYGÔNES(RECTANGLESCOMPRIS): '
      READ(*,*)NBR

      ... Pour chaque polygône ...

      DO 3 K=1,NBR
       WRITE(*,*)'QUEL EST LE NOMBRE DE POINTS POUR LE RECTANGLE ',K
       READ(*,*)N
       WRITE(*,*)'VEUILLEZ INTRODUIRE LE RECTANGLE No ',K,' A REMPLIR.'
       WRITE(*,*)  'AE'
       READ(*,*) AE
       WRITE(*,*)  'BE'
       READ(*,*) BE
       WRITE(*,*)  'CE'
       READ(*,*) CE
       WRITE(*,*)  'DE'
       READ(*,*) DE

         ... Si on veut un polygône et pas le rectangle...

       WRITE(*,*)'NOMBRE DE DROITES POUR OBTENIR UN POLYGÔNE'
       READ(*,*)NDR
       DO 4 J=1,NDR
          WRITE(*,*)'LES COÉFFICIENTS DE LA DROITE ',J, ' aX+bY+c=0 SONT:'
          WRITE(*,*)'a='
          READ(*,*)coefa(J)
          write(*,*)'b='
          read(*,*)coefb(J)
          write(*,*)'c='
          read(*,*)coefc(J)
          WRITE(*,*)'CÔTE POSITIF[1] OU NEGATIF[2] ? '
          READ(*,*)POSI(J)
       CONTINUE
4

       CALL RNJPT(3)
       ISEED = 123457
       CALL RNSET(ISEED)

       ... On génère les N points dans le rectangle [AE,BE] [CE,BE]...

       DO  5   I=1,N

           Y(I) = RNUNF()

           X(I,1) = (Y(I)*(BE-AE))+AE

           Z(I) = RNUNF()

           X(I,2) = (Z(I)*(DE-CE))+CE

       CONTINUE
5

       ... On prend la partie entière des points générés ...

       DO  10  I=1,N
           DO  15  J=1,2
               R(I,J) = INT(X(I,J))
15         CONTINUE
10       CONTINUE

       IF (NDR.EQ.0) THEN

          ... Si on veut un rectangle, on écrit ces N points dans le fichier.

          WRITE(7,11,ERR=9) ((R(I,J),J=1,2),I=1,N)
11        FORMAT((2I))
          WRITE(*,*)'LE RECTANGLE ',K,' A',N,' POINTS'
       ELSE
```

```
C
C        ... Sinon, pour chaque point, on vérifie les équations de droites.
C
         G = 0
         DO 17 I=1,N
            GARDE = .TRUE.
            L=1
20          IF (GARDE.AND.(L.LE.NDR)) THEN
C
               EQUA = COEFA(L)*R(I,1)+COEFB(L)*R(I,2)+COEFC(L)
               IF (POSI(L).EQ.1) THEN
                  IF (EQUA.LT.0) THEN
                     GARDE = .FALSE.
                  ENDIF
               ELSE
                  IF (EQUA.GT.0) THEN
                     GARDE = .FALSE.
                  ENDIF
               ENDIF
               L = L+1
               GOTO 20
            ENDIF
C
C           ...Si on garde le point, on l'écrit dans le fichier...
C
            IF (GARDE) THEN
               WRITE(7,11,ERR=9) (R(I,J),J=1,2)
               G = G+1
            ENDIF
C
C           ... sinon, on passe au point suivant
C
17       CONTINUE
         WRITE(*,*)'LE POLYGONE',K,' A',G,' POINTS'
      ENDIF
C
C     CONTINUE
C
C
C
      WRITE(*,*) ' QUEL EST LE NOMBRE D''ELLIPSES :'
      READ(*,*) NBE
C
      DO 25  K=1,NBE
C
C        ... pour chaque ellipse ...
C
         WRITE(*,*)'QUEL EST LE NOMBRE DE POINTS POUR L''ELLIPSE ',K
         READ(*,*)N
         WRITE(*,*)' VEUILLER INTRODUIRE L''ELLIPSE No ',K,'A REMPLIR'
         WRITE(*,*)' A :'
         READ(*,*)A
         WRITE(*,*)' B :'
         READ(*,*)B
         WRITE(*,*)' C1 :'
         READ(*,*)C1
         WRITE(*,*)' C2 :'
         READ(*,*)C2
C
C        ... on génère les points dans le rectangle [C1-A,C1+A] [C2-B,C2+B]..
C
         AE = C1 - A
         BE = C1 + A+1
         CE = C2 - B
         DE = C2 + B+1
C
         CALL RNOPT(3)
         ISEED = 123457
         CALL RNSET(ISEED)
C
```

```
C
      DO  30   I=1,N
C
        Y(I) = RNUNF()
C
        X(I,1) = (Y(I)*(BE-AE))+AE
C
        Z(I) = RNUNF()
C
        X(I,2) = (Z(I)*(DE-CE))+CE
C
30    CONTINUE
C
C
      DO  35  I=1,N
          DO  40  J=1,2
              R(I,J) = INT(X(I,J))
40        CONTINUE
35    CONTINUE
C
      G = 0
      DO 45  I=1,N
C
C        ... Pour chaque point, on vérifie s'il est à l'intérieur ...
C                                    de l'ellipse
C
         EQUA = ((R(I,1)-C1)**2/A**2) + ((R(I,2)-C2)**2/B**2)
         IF (EQUA.LE.1.0D0) THEN
         WRITE(7,11,ERR=9) (R(I,J),J=1,2)
         G =G+1
         ENDIF
45    CONTINUE
      WRITE(*,*)'L''ELLIPSE ',K,' A',G,' POINTS'
C
25  CONTINUE
C
    END
C
C
C
```

Appendix B

Algorithme basé sur les hyperplans

Les listings proposés dans cette section concernent le programme décrit dans le chapitre 5.

Nous présentons tout d'abord le programme principal et ensuite, ses sous-routines.

Pour chacune des routines, nous spécifions les paramètres ainsi que les variables locales.

```
PROGRAM CLASAUTOVP

*****************************************************************************
*                                                                           *
*    BUT DU PROGRAMME :                                                     *
*    ------------------                                                     *
*                                                                           *
*    Cet algorithme en classification automatique est divise en             *
*    trois parties.                                                         *
*                                                                           *
*          ## Recherche des points du convexe de depart. (points            *
*             extremaux dans le repere canonique et dans celui              *
*             des vecteurs propres.)                                        *
*                                                                           *
*          #* Division de celui-ci par les hyperplans separateurs.         *
*                                                                           *
*          ** Recollement de ces classes.                                   *
*                                                                           *
*    Il utilise les procedures suivantes :                                  *
*                                                                           *
*      . DEVCSF et DLINRG dans IMSL/LIB                                     *
*      . INIT , ISECOND , ISECONDIFF dans _ML:UTIL/LIB                      *
*      . CONVEXE dans le fichier CONVEXE_NEW.FOR                            *
*                compilation : BLAS/LIB,_ML:LINPACK/LIB                     *
*                                                                           *
*    Il est efficace uniquement pour les problemes bien structures          *
*    et en dimension deux.                                                  *
*                                                                           *
*****************************************************************************

TABLE DES CONSTANTES:
---------------------

NPMAX : Constante entiere qui contient le nombre maximal de
        points du probleme .

DIMMAX : Constante entiere qui contient la dimension maximale
         du probleme .

CLAMAX : Constante entiere qui contient le nombre maximal
         de classes .

CLIMAX : Constante entiere qui contient le nombre maximal
         de classes avant le recollement.


TABLE DES VARIABLES DU PROGRAMME:
---------------------------------

UL : Variable entiere qui contient le numero d'unite logique
     associee au fichier de donnees .

DIM : Variable entiere qui contient la dimension du probleme .

IOS : Variable entiere qui permet de gerer les erreurs
      d'entrees/sorties .

FICDON : Variable caractere de longueur 20 qui contient le
         nom du fichier de donnees .

PTDEB : Tableau d'entiers de dimension NPMAX sur DIMMAX
        qui contient les coordonnees des differents points du
        probleme . Les points (au nombre de NP) appartiennent
        a un espace de dimension DIM .

NP: Variable entiere qui contient le nombre de points
    du probleme .

US : Variable entiere qui contient le numero d'unite logique
     associee au fichier contenant les resultats.
```

FICRES : Variable caractere de longueur 20 qui contient le nom du fichier qui contiendra les resultats .

NBRCLA : Variable entiere qui contient le nombre de classe desire .

I,J,K,R : Variables entieres de boucle .

POINT : Tableau de double precision de dimension NPMAX sur DIMMAX qui contient les coordonnees des points du probleme translate par rapport au centre de symetrie .

VECPR : tableau d.p. de dimension dimmax sur dimmax qui contient les dimmax composantes des dimmax vecteurs propres de MAT.

NBC : Variable entiere qui contient le nombre de classes courant.

NBPC : Vecteur d'entiers de dimension CLIMAX qui contient le nombre de points associe a chaque classe.

VOLTOT : Variable de double precision qui contient le volume de convexe de depart .

PTF : Tableau d'entiers de dimension DIMMAX sur NPMAX qui contient les coordonnees des differents points dont il faut chercher l'enveloppe convexe. Les points (au nombre de NP) appartiennent a un espace de dimension DIM. Ils sont donc caracterises par DIM entiers (DIM <= DIMMAX).

HYPERPLAN : Tableau de double precision de dimension DIMMAX sur NHMAX qui contient les coefficients des equations des hyperplans. A la sortie de la routine, il y aura NH hyperplans. Les equations seront de la forme :

$$\sum_{i=1,dim} (alpha_i * x_i) = 1 \text{ ou } 0$$

Un hyperplan, dans un espace de dimension DIM, est donc caracterise par dim coefficients 'alpha' et un terme independant. Le terme independant se trouvera,a la sortie, dans la premiere ligne du tableau HYPER1.

Les coefficients 'alpha' seront dans le tableau HYPERPLAN. Les 'x' correspondent aux coordonnees d'un point. Le terme independant peut etre 1 ou 0 suivant que l'hyperplan passe ou non par l'origine (0,...,0).

VECTRA : Vecteur de double precision de dimension DIMMAX . Il s'agit d'un vecteur de translation utilise pour que le point de coordonnees (0,0,...,0) soit interieur au polyedre de base. Ceci facilite la recherche des equations des hyperplans. Si COND est 'true', les equations des hyperplans seront relatives a ce vecteur. Sinon, elles seront absolues (i.e. relatives a la vraie origine (0,...,0)).

FACE : Tableau d'entiers de dimension DIMMAX sur NHMAX qui contient les references des points composants une face. Cette reference correspond au tableau POINT. Il faut faire tres attention au fait que pour nous, une face est un ensemble de DIM points differents. Par exemple :

- en dimension 1 : face = un point
- en dimension 2 : face = un segment
- en dimension 3 : face = un triangle
- en dimension 4 : face = un tetraedre
- ...

De ce fait, un meme hyperplan peut contenir plusieurs faces.

POINT1 : Vecteur d'entiers de dimension NPMAX qui contient les statuts associes a chaque point. On considere qu'il y a trois statuts possibles durant l'execution de la sous-routine :

. -1 : le point n'a pas encore ete traite.
. 0 : le point est interne au polyedre courant.
. +1 : le point est sommet du polyedre courant.

A la sortie de la sous-routine, seuls les deux derniers statuts restent actifs (i.e. on associe a chaque point un des deux derniers statuts).

HYPER1 : Tableau d'entiers de dimension 2 qui contient dans la premiere ligne les statuts associes a chaque hyperplan. On considere qu'il y a trois statuts possibles :

- 0 : l'hyperplan est interne au polyedre courant.
- 1 : l'hyperplan est transitoire.
- 2 : l'hyperplan est final.

Le deuxieme ligne du tableau contient le nombre de faces inclues dans un hyperplan. La troisieme ligne contient la reference de la premiere face inclue dans l'hyperplan. Cette reference correspond au tableau FACE . A la sortie de la sous-routine, seuls les hyperplans finaux sont conserves. Des lors, la premiere ligne de HYPER1 n'a plus de raison d'etre. On y met le terme independant de l'equation de l'hyperplan.

FACE1 : Vecteur d'entiers. Dans ce vecteur , a chaque face correspond un nombre. Il correspond a la reference d'une face appartenant au meme hyperplan. S'il n'y en a pas d'autre, sa valeur est 0.

NBPF : Variable entiere contenant le nombre de points dont il faut chercher l'enveloppe convexe.

NH : Variable entiere contenant le nombre courant d'hyperplans. A la sortie, cette variable contiendra le nombre d'hyperplans finaux.

NF : Variable entiere contenant le nombre courant de faces. A la sortie, cette variable contiendra le nombre de faces inclues dans les hyperplans finaux.

VOLUME : Variable double precision contenant le volume du polyedre.

CONV : Tableau d'entiers de dimension CLAMAX sur NPMAX dont la case (I,K) contient le numero du K eme point de la classe I .

T1 : Temps CPU pour trouver les sommets du convexe de depart.

T2 : Temps CPU pour la division de celui-ci par les hyperplans separateurs .

T3 : Temps CPU pour le recollement des classes.

NBH : Nombre d'hyperplans separateur qui diviseront le convexe de depart.

IMIN : Tableau d'entiers qui contient les indices des points minimum de chaque dimension.

IMAX : Tableau d'entiers qui contient les indices des points maximum de chaque dimension.

IMINRP : Tableau d'entiers qui contient les indices des points minimum de chaque dimension dans le repere des vecteurs propres.

IMAXRP : Tableau d'entiers qui contient les indices des points maximum de chaque dimension dans le repere des vecteurs propres.

NB : Nombre de points extremaux differents.

NSOM : Nombre de sommets du convexe de depart.

IEX : Tableau d'entiers contenant les indices des points extremaux differents (c'est une synthese des tableaux IMIN,IMAX,IMINRP, IMAXRP).

M :

NBTOT : var. entiere qui contient le nombre total de points dans les classes avant le recollement.(pour la verification)

IND : var. entiere qui represente l'indice dans le nouveau convexe du point que l'on traite s'il y a lieu de le mettre dans celui-ci

A,B : var. entieres qui representent les extremites d'un cote du convexe de depart.

NBPR : var. entiere qui contient le nombre de points dans la classe RESTE.

RESTE : tableau d.p. qui contient les coordonnees des points qui ne sont pas dans le convexe de depart.

SOM : variable d.p. intermediaire pour calculer le centre de gravite.

CG : vecteur d.p. de longueur DIM qui donne les coordonnees du centre de gravite.

EGAL : var. logique qui est a false si le point extremum que l'on est en train de traiter n'est pas deja dans le vecteur IEX

INTER : tableau de variable logique qui positionne le point (o,o) par rapport aux cotes du convexe.

BOOL : tableau de variable logique qui positionne le point considere par rapport aux cotes du convexe.

DIFF : var. logique qui est a true s'il y a une difference entre les tableaux INTER et BOOL.

COND :

AUX1,AUX2 : var entieres qui representent les deux classes que l'on recolle.

PTMIN : vecteur d.p. qui contient les DIM valeurs minimales (une par direction).

PTMAX : vecteur d.p. qui contient les DIM valeurs maximales (une par direction).

PTREPROMIN : vecteur d.p. qui contient les DIM valeurs minimales (une par direction) dans le repere des vecteurs propres.

PTREPROMAX : vecteur d.p. qui contient les DIM valeurs maximales (une par direction) dans le repere des vecteurs propres.

PTREPRO : tableau d.p. qui contient les DIM coordonnees des points dans le repere des vecteur propres.

VARPR : vecteurs d.p. qui contient les DIM valeurs propres de la matrice MAT.

INVVECTPR : matice d.p. inverse de la matrice des vecteurs propres.

VOLU : vecteur d.p. qui contient le volume de chaque classe.

SOMVP : tableau d.p. qui contient les coordonnees dans la base canonique des points extremaux dans le repere des vect. pr.

HYPT : a chaque ligne ,on a :
colonne 1 et 2 : les extremites du premier cote
" " 3 et 4 : " " " du second cote
colonne 5 : le terme independant de l'hyperplan separateur.
" 6 : le coefficient de x " " "
colonne 7 : " " " y

MAT : matrice d.p. resultat du produit de la matrice des points par sa transposee.

EQUA : var d. p. pour rendre l'ecriture plus compacte.

```
C
C     DECLARATION DES VARIABLES:
C     --------------------------
C
      INTEGER DIMMAX,CLIMAX,CLAMAX,NPMAX,NHMAX
      PARAMETER (DIMMAX=4,CLIMAX=30,CLAMAX=15,NPMAX=1000,NHMAX=1000)
C
      INTEGER  IOS,DIM,NP,I,UL,NBRCLA,US,J,K,R,T1,T2,NBH,NBTOT,
     +         IMIN(DIMMAX),IMAX(DIMMAX),IMINRP(DIMMAX),NBC,IND,G,H,
     +         IMAXRP(DIMMAX),NSOM,NB,ISECONDIFF,IEX(4*DIMMAX),T3,
     +         A,B,NBPC(CLIMAX),NBPR,CONV(CLIMAX,NPMAX),RESTE(NPMAX),
     +         PTDEB(NPMAX,DIMMAX),AUX1,AUX2,NBPF,NF,NH,FACE1(NHMAX),
     +         POINT1(NPMAX),PTF(DIMMAX,NPMAX),FACE(DIMMAX,NHMAX),
     +         HYPER1(3,NHMAX),L,AUG,NOMBRE
C
      DOUBLE PRECISION POINT(NPMAX,DIMMAX),SOM(DIMMAX),CG(DIMMAX),
     +         PTMIN(DIMMAX),PTMAX(DIMMAX),VAPR(DIMMAX),
     +         VECTPR(DIMMAX,DIMMAX), INVVECTPR(DIMMAX,DIMMAX),
     +         PTREPRO(NPMAX,DIMMAX),PTREPROMIN(DIMMAX),VOLU(CLIMAX),
     +         PTREPROMAX(DIMMAX),SOMVP(2*DIMMAX,DIMMAX),VECTRA(DIMMAX),
     +         SOMMET(4*DIMMAX,DIMMAX),HYPT(20,7),SOMEX(4*DIMMAX,DIMMAX),
     +         MAT(DIMMAX,DIMMAX),EQUA,HYPERPLAN(DIMMAX,NHMAX),
     +         VOLUME,MIN,ECART(CLIMAX,CLIMAX),criter
C
      CHARACTER *20 FICDON,FICRES
C
      LOGICAL EGAL,INTER(8),BOOL(8),DIFF,COND,LOGIK
C
C
      EXTERNAL INIT,ISECOND,ISECONDIFF,DEVCSF,COTE,DIVISIONDIM2,ORDRE,
     +         CONVEXE,DLINRG
```

LE PROGRAMME

```
      CALL INIT
C     ---------------------------------------------
C     ... lecture du nom du fichier de donnees ...
C     ---------------------------------------------
C
      WRITE(*,10)
10    FORMAT(' Avec quel fichier de donnees voulez-vous travailler ? ')
      READ(*,15)FICDON
15    FORMAT(A20)
      UL = 30
C     ---------------------------------------
C     ... ouverture du fichier de donnees ...
C     ---------------------------------------
C
      OPEN(UNIT=UL,FILE=FICDON,STATUS='OLD',ERR=20,IOSTAT=IOS)
20    IF (IOS.GT.0) THEN
         WRITE(*,25)
25       FORMAT(' ERREUR D''OUVERTURE, CE FICHIER N''EXISTE PAS. ')
         GOTO 5
      ENDIF
C     -----------------------------------------
C     ...lecture de la dimension du probleme ...
C     -----------------------------------------
C
30    CONTINUE
      WRITE(*,35)
35    FORMAT(' Quelle est la dimension du probleme ? ')
      READ(*,36)DIM
36    FORMAT(I2)
      IF ((DIM.LE.0).OR.(DIM.GT.DIMMAX)) THEN
         WRITE(*,*)' DIMENSION INCORRECTE, VEUILLEZ RETAPER. '
         GOTO 30
      ENDIF
C
```

```
C     ------------------------------------------------------
C     ... lecture des donnees + calcul du nombre de points ...
C     ------------------------------------------------------

      NP = 0
40    CONTINUE
         READ(UL,*,END=45) (PTDEB(NP+1,I),I=1,DIM)
         NP = NP+1
         GOTO 40
45    CONTINUE
C     ---------------------------------------
C     ... fermeture du fichier de donnees ...
C     ---------------------------------------

      CLOSE (UL)

C     -------------------------------------
C     ... demande du nombre de classes ...
C     -------------------------------------

50    CONTINUE
      WRITE(*,55)
55    FORMAT(' Combien de classes desirez-vous a la fin ? ')
      READ(*,*)NBRCLA
      IF ((NBRCLA.LE.0).OR.(NBRCLA.GT.CLAMAX)) THEN
         WRITE(*,*)' NOMBRE INCORRECT, VEUILLEZ RETAPER. '
         GOTO 50
      ENDIF
C     -----------------------------------------
C     ... lecture du nom du fichier resultat ...
C     -----------------------------------------

      WRITE(*,70)
70    FORMAT(' Dans quel fichier voulez-vous mettre les resultats ?')
      READ(*,75)FICRES
75    FORMAT(A20)
      US=20
C
      CALL ISECOND(T1)

C     ------------------------------------------
C     ... determination du centre de gravite ...
C           et translation des points
C     ------------------------------------------


      DO 80  J=1,DIM
         SOM(J) = 0
         DO 85  I=1,NP
            SOM(J) = SOM(J) + PTDEB(I,J)
85       CONTINUE
80    CONTINUE
C
      DO 90  J=1,DIM
         CG(J) = SOM(J) / NP
C         WRITE(*,*)  ' CG :',CG(J)
         DO 95  I=1,NP
            POINT(I,J) = PTDEB(I,J) - CG(J)
95       CONTINUE
90    CONTINUE
C     ------------------------------------------
C     ...recherche des 2*DIM points extremaux ...
C     ------------------------------------------


      DO 100  J=1,DIM
         PTMIN(J) = 0
         PTMAX(J) = 0
         DO 105  I=1,NP
            IF (POINT(I,J).LE.PTMIN(J)) THEN
               IMIN(J) = I
               PTMIN(J) = POINT(I,J)
            ENDIF
            IF (POINT(I,J).GE.PTMAX(J)) THEN
               IMAX(J) = I
               PTMAX(J) = POINT(I,J)
            ENDIF
105      CONTINUE
100   CONTINUE
C
```

```
C     ------------------------------------------
C     ... recherche des DIM vecteurs propres ...
C     ------------------------------------------

      DO 110  I=1,DIM
         DO 115  J=1,DIM
            MAT(I,J) = 0
            DO 120  K=1,NP
               MAT(I,J) = MAT(I,J) + (POINT(K,I)*POINT(K,J))
120         CONTINUE
115      CONTINUE
110   CONTINUE
C
      CALL DEVCSF(DIM,MAT,DIMMAX,VAPR,VECTPR,DIMMAX)
C
C
C
C     ... recherche des 2*DIM points extremaux ...
C         dans le repere des vecteurs propres
C     ------------------------------------------
C
C     # #  inversion de la matrice des vecteurs propres  ##
C
      CALL DLINRG(DIM,VECTPR,DIMMAX,INVVECTPR,DIMMAX)
C
C     # #  coordonnees des points dans le repere des vecteurs propres
C
      DO 125  I=1,NP
         DO 130  J=1,DIM
            PTREPRO(I,J) = 0
            DO 135  K=1,DIM
               PTREPRO(I,J) = PTREPRO(I,J)+(INVVECTPR(J,K)*POINT(I,K))
135         CONTINUE
130      CONTINUE
125   CONTINUE
C
C     # #  recherche des points extremaux dans cette base  # #
C
      DO 140  J=1,DIM
         PTREPROMIN(J) = 0
         PTREPROMAX(J) = 0
         DO 145  I=1,NP
            IF (PTREPRO(I,J).LE.PTREPROMIN(J)) THEN
               IMINRP(J) = I
               PTREPROMIN(J) = PTREPRO(I,J)
            ENDIF
            IF (PTREPRO(I,J).GE.PTREPROMAX(J)) THEN
               IMAXRP(J) = I
               PTREPROMAX(J) = PTREPRO(I,J)
            ENDIF
145      CONTINUE
140   CONTINUE
C
C
C     ------------------------------------------------------------
C     ... coordonnees de ces points dans la 1ere base (naturelle) ...
C     ------------------------------------------------------------
C
      DO 150  I=1,DIM
         DO 155  J=1,DIM
            SOMVP(I,J) = 0
            DO 160  K=1,DIM
               SOMVP(I,J) = SOMVP(I,J)+
     +                      (VECTPR(J,K)*PTREPRO(IMINRP(I),K))
160         CONTINUE
155      CONTINUE
150   CONTINUE
C
      DO 165  I=DIM+1,2*DIM
         DO 170  J=1,DIM
            SOMVP(I,J) = 0
            DO 175  K=1,DIM
               SOMVP(I,J) = SOMVP(I,J)+
     +                      (VECTPR(J,K)*PTREPRO(IMAXRP(I-DIM),K))
175         CONTINUE
170      CONTINUE
165   CONTINUE
```

```
C
      T1 = ISECONDIFF(T1)
C
C     ------------------------------------------------
C     ... prendre en compte les sommets distincts ...
C              dans imax,imin,imaxrp,iminrp
C     ------------------------------------------------
C
      I = 1
C
C
      IEX(I) = IMAX(I)
      DO 180  K=2,DIM
         R = 1
         EGAL = .FALSE.
185      IF ((EGAL.EQ..FALSE.).AND.(R.LE.I)) THEN
            IF (IMAX(K).EQ.IEX(R)) THEN
               EGAL = .TRUE.
            ENDIF
            R = R + 1
            GOTO 185
         ENDIF
         IF (EGAL.EQ..FALSE.) THEN
C
            I = I+1
            IEX(I) = IMAX(K)
         ENDIF
180   CONTINUE
C
C
      DO 190  K=1,DIM
         R = 1
         EGAL = .FALSE.
195      IF ((EGAL.EQ..FALSE.).AND.(R.LE.I)) THEN
            IF (IMIN(K).EQ.IEX(R)) THEN
               EGAL = .TRUE.
            ENDIF
            R = R + 1
            GOTO 195
         ENDIF
         IF (EGAL.EQ..FALSE.) THEN
            I = I+1
            IEX(I) = IMIN(K)
         ENDIF
190   CONTINUE
C
      DO 200  K=1,DIM
         R = 1
         EGAL = .FALSE.
205      IF ((EGAL.EQ..FALSE.).AND.(R.LE.I)) THEN
            IF (IMAXRP(K).EQ.IEX(R)) THEN
               EGAL = .TRUE.
            ENDIF
            R = R + 1
            GOTO 205
         ENDIF
         IF (EGAL.EQ..FALSE.) THEN
            I = I+1
            IEX(I) = IMAXRP(K)

         ENDIF
200   CONTINUE
```

```
      DO 210  K=1,DIM
         R = 1
         EGAL = .FALSE.
215      IF ((EGAL.EQ..FALSE.).AND.(R.LE.I)) THEN
            IF (IMINRP(K).EQ.IEX(R)) THEN
               EGAL = .TRUE.
            ENDIF
            R = R + 1
            GOTO 215
         ENDIF
         IF (EGAL.EQ..FALSE.) THEN
            I = I+1
            IEX(I) = IMINRP(K)
         ENDIF
210   CONTINUE
C
      NB = I

      DO 220  I=1,NB
         DO 225  J=1,DIM
            SOMEX(I,J) = POINT(IEX(I),J)
225      CONTINUE
220   CONTINUE
C
C
C     ------------------------------------------
C     ... mettre les sommets en ordre cyclique ...
C     ------------------------------------------
C
      CALL ORDRE(SOMEX,NB,DIM,DIMMAX,COTE,SOMMET,NSOM)
C
C
      CALL ISECOND(T2)
C     ------------------------------
C     ... division en dimension 2 ...
C     ------------------------------
C
      CALL DIVISIONDIM2(NSOM,SOMMET,DIMMAX,HYPT,NBH)
C
C
      T2 = ISECONDIFF(T2)
C     ------------------------------------------------------
C     ... mettre les premiers resultats dans le fichier ...
C     ------------------------------------------------------
C
      OPEN (UNIT=US,FILE=FICRES,STATUS='NEW')
      WRITE(US,*)' NOMBRE TOTAL DE POINTS DU PROBLEME = ',NP
C
      WRITE(US,*)' LE CENTRE DE GRAVITE : ',(CG(J),J=1,DIM)
C
      WRITE(US,*)' LES',DIM,' VECTEURS PROPRES : ',
     +              ((VECTPR(I,J),I=1,DIM),J=1,DIM)

      WRITE(US,*)' LES ',NSOM,' SOMMETS EXTREMAUX SONT :'
      DO 230  I=1,NSOM
         DO 235  J=1,DIM
            WRITE(US,*) SOMMET(I,J)
235      CONTINUE
         WRITE(US,*)
230   CONTINUE
C
      WRITE(US,*)' LES ',NBH,' HYPERPLANS SEPARATEURS'
      DO 240  I=1,NBH
         WRITE(US,*)' COTE AB :',HYPT(I,1),HYPT(I,2)
         WRITE(US,*)' COTE CD :',HYPT(I,3),HYPT(I,4)
         WRITE(US,*)' COEFF: 0',HYPT(I,5),' 1',HYPT(I,6),' 2',HYPT(I,7)
         WRITE(US,*)
240   CONTINUE
C
      WRITE(US,*)' TEMPS DE RECHERCHE DES POINTS ',T1
      WRITE(US,*)' TEMPS DE RECHERCHE DES HYPERPLANS ',T2
```

```
C     ---------------------------------------
C     ... Recherche du convexe de depart ...
C     ---------------------------------------
C
      CALL ISECOND(T3)
C
C     # position du centre de gravite (o,o) par rapport aux cotes #
C
      DO 245  A=1,NSOM
         IF (A.EQ.NSOM) THEN
            B = 1
         ELSE
            B = A+1
         ENDIF
         EQUA = -(SOMMET(B,1)-SOMMET(A,1))*SOMMET(A,2)
     +          +(SOMMET(B,2)-SOMMET(A,2))*SOMMET(A,1)
         INTER(A) = (EQUA.LE.0)
C
245   CONTINUE
C
      NBPC(1) = 0
      NBPR = 0
C
C     # position de chaque point par rapport a (o,o) #
C
      DO 255  I=1,NP
         DO 250  A=1,NSOM
            IF (A.EQ.NSOM) THEN
               B = 1
            ELSE
               B = A+1
            ENDIF
            EQUA = (SOMMET(B,1)-SOMMET(A,1))*(POINT(I,2)-SOMMET(A,2))
     +             -(SOMMET(B,2)-SOMMET(A,2))*(POINT(I,1)-SOMMET(A,1))
            BOOL(A) = (EQUA.LE.0)
250      CONTINUE
         DIFF = .FALSE.
         J = 0
C
C        # si le point est situer comme (o,o) alors il est a l'interieur
C
256      IF ((DIFF.EQ..FALSE.).AND.(J.LE.NSOM)) THEN
            IF (BOOL(J).EQ.INTER(J)) THEN
               J = J+1
            ELSE
               DIFF = .TRUE.
            ENDIF
            GOTO 256
         ENDIF
         IF (J.EQ.NSOM+1) THEN
            NBPC(1) = NBPC(1) + 1
            CONV(1,NBPC(1)) = I
         ELSE
            NBPR = NBPR + 1
            RESTE(NBPR) = I
         ENDIF
255   CONTINUE
C
C     ---------------------------------------
C     ... Recherche des differentes classes ...
C     ---------------------------------------
C
      NBC = 1
C
      DO 260  I=1,NBH
C                    ... pour chaque hyperplan ...
C
         AUG = 0
         J = 1
265      IF (J.LE.NBC) THEN
C                       ... pour chaque classe ...
C
```

```
              IND = 0
              K = 0
              IF (NBPC(J).EQ.0) THEN
C                WRITE(*,*)'LA CLASSE No',j,'EST VIDE'
                 DO 267  L=J,NBC-1
                    DO 268  G=1,NBPC(L+1)
                       CONV(L,G) = CONV(L+1,G)
268                 CONTINUE
                    NBPC(L) = NBPC(L+1)
267              CONTINUE
                 NBC = NBC-1
              ELSE
                 NOMBRE = NBPC(J)
270              IF (K.LT.NBPC(J)) THEN
C                   ... tant que l'on n'a pas classe tous les points ...
C
C
                    K = K+1
                    EQUA = HYPT(I,5)+POINT(CONV(J,K),1)*HYPT(I,6)
     +                    + POINT(CONV(J,K),2)*HYPT(I,7)
C
                    IF (EQUA.LE.0) THEN
                       IND = IND+1
                       CONV(NBC+1+AUG,IND) = CONV(J,K)
                       DO 275  R=K,NBPC(J)-1
                          CONV(J,R) = CONV(J,R+1)
275                    CONTINUE

                       NBPC(J) = NBPC(J)-1
                       K = K-1
                    ENDIF
                    GOTO 270
                 ENDIF
                 IF ((NBPC(J).NE.0).AND.(NBPC(J).NE.NOMBRE)) THEN
                    NBPC(NBC+1+AUG) = IND
                    AUG = AUG+1
                    J = J+1
                 ELSE
                    IF (NBPC(J).EQ.0) THEN
                       DO 275  L=1,IND
                          CONV(J,L) = CONV (NBC+1+AUG,L)
275                    CONTINUE
                       NBPC(J) = IND
                       J = J+1
                    ELSE
C                      (NBPC(J).EQ.NOMBRE)
                       J = J+1
                    ENDIF
C
                 ENDIF
C
              ENDIF
C
C
              GOTO 265
           ENDIF
           NBC = NBC+AUG
260     CONTINUE
C
        T3 = ISECONDIFF(T3)
C
        DO G =1,NBC
           DO H =1,NBPC(G)
              WRITE(*,*)'CONV',G,':',CONV(G,H)
        ENDDO
        ENDDO
C
C       ------------------------------------------
C       ... mettre les resultats dans le fichier ...
C       ------------------------------------------
C
        WRITE(*,*)
        WRITE(US,*)
        WRITE(US,*)'LES ',NBC,' CLASSES ONT COMME POINTS :'
        WRITE(US,*)
        DO 280  I=1,NBC
           WRITE(US,*)
```

```
      WRITE(US,*)
      WRITE(US,*)' LA CLASSE No ',I
      DO 285  J=1,NBPC(I)
         WRITE(US,*)' LE POINT No ',CONV(I,J),' DE COORD.: ',
     +                (PTDEB(CONV(I,J),K),K=1,DIM)
285   CONTINUE
      WRITE(US,*)' LA CLASSE ',I,' COMPTE ',NBPC(I),' POINTS'
      WRITE(US,*)
280   CONTINUE
C
      NBTOT = 0
      DO 295 I=1,NBC

         NBTOT = NBTOT + NBPC(I)
295   CONTINUE
C
      WRITE(US,*)NBTOT,'+',NBPR,'=',NP
C
      WRITE(US,*)
      WRITE(US,*)' LA CLASSE RESTE :'
      DO 290  I=1,NBPR
         WRITE(US,*)'LE POINT No',RESTE(I),'DE COORD. :',
     +              (PTDEB(RESTE(I),J),J=1,DIM)
290   CONTINUE
C
      WRITE(US,*)
      WRITE(US,*)' TEMPS DE RECHERCHE DES CLASSES :',T3
C
C     ------------------------------------------------------
C     ... recherche de l'enveloppe convexe de chaque classe ...
C     ------------------------------------------------------
C
C
      COND = .true.
C
      DO 296  I=1,NBC
         DO 300  J=1,NBPC(I)
            DO 305  K=1,DIM
               PTF(K,J) = PTDEB(CONV(I,J),K)
305         CONTINUE
300      CONTINUE
         WRITE(*,*)'ENV. CONV. DE LA CLASSE',I
         CALL CONVEXE(PTF,HYPERPLAN,VECTRA,FACE,POINT1,HYPER1,
     +                FACE1,DIM,NBPC(I),NH,NF,VOLUME,DIMMAX,COND)
         VOLU(I) = VOLUME
C         WRITE(*,*)'SON VOLUME :',VOLUME
C
296   CONTINUE
C
C     .... recherche de l'enveloppe convexe de la classe RESTE ...
C
      IF (NBPR.NE.0) THEN
         DO 310  J=1,NBPR
            DO 315  K=1,DIM
               PTF(K,J) = PTDEB(RESTE(J),K)
315         CONTINUE
310      CONTINUE
C
         WRITE(*,*)'ENV. CONV. DE LA CLASSE RESTE'
         CALL CONVEXE(PTF,HYPERPLAN,VECTRA,FACE,POINT1,HYPER1,
     +                FACE1,DIM,NBPR,NH,NF,VOLUME,DIMMAX,COND)
C
C     ------------------------------------------------
C     ... mettre la classe reste comme derniere classe ...
C     ------------------------------------------------
         NBC = NBC+1
         DO 316  J=1,NBPR
            CONV(NBC,J) = RESTE(J)
316      CONTINUE
C
         NBPC(NBC)=NBPR
         VOLU(NBC) = VOLUME
      ENDIF
```

```
C
C       WRITE(*,*)'SON VOLUME : ',VOLUME
C
C     ------------------
C     ... RECOLLEMENT ...
C     ------------------
C
      DO 320  J=1,NBC
         DO 325  I=1,J-1
C
C           ... on met dans nbpf le nombre de points contenus ...
C               dans la classe obtenue par la fusion de deux autres
C
            NBPF = NBPC(I) + NBPC(J)
C
C           ... stockage des coordonnees des points dans ptf ...
C
            DO 330  R=1,NBPC(I)
               DO 335  K=1,DIM
                  PTF(K,R) = PTDEB(CONV(I,R),K)
335            CONTINUE
330         CONTINUE
C
            DO 340  R=1,NBPC(J)
               DO 345  K=1,DIM
                  PTF(K,R+NBPC(I)) = PTDEB(CONV(J,R),K)
345            CONTINUE
340         CONTINUE
C
C           ... calcul de l'enveloppe convexe de la classe ...
C            obtenue par la fusion de deux autres classes
C
            WRITE(*,*)'FUSION DE',J,I
C
            CALL CONVEXE(PTF,HYPERPLAN,VECTRA,FACE,POINT1,HYPER1,
     +                   FACE1,DIM,NBPF,NH,NF,VOLUME,DIMMAX,COND)
C
C           ... calcul de l'ecart entre les deux classes considerees ...
C
            ECART(I,J) = VOLUME - (VOLU(I)+VOLU(J))
            WRITE(*,*)'ECART ENTRE',J,'ET',I,':',ECART(I,J)
C
325      CONTINUE
320   CONTINUE
C
C     ------------------------------------------------------
C     ... tant qu'on n'a pas le nombre de classes desire ...
C     ------------------------------------------------------
C
350   IF (NBC.GT.NBRCLA) THEN
C
C        ... recherche des numeros des deux classes entre lesquelles ...
C                    l'ecart est minimal
C
         WRITE(*,*)'NBC',NBC
         MIN = 1.D+12
         DO 355  J=1,NBC
C
            DO 360  I=1,J-1
               IF (ECART(I,J).LT.MIN) THEN
                  MIN = ECART(I,J)
                  AUX1 = I
                  AUX2 = J
                  WRITE(*,*)'AUX : ',AUX1,AUX2
               ENDIF
360         CONTINUE
355      CONTINUE
C
C        ... on remplit la matrice des points contenus dans la fusion
C                          des classes AUX1 et AUX2
C
         DO 365  R=1,NBPC(AUX1)
            DO 370  K=1,DIM
               PTF(K,R) = PTDEB(CONV(AUX1,R),K)
370         CONTINUE
365      CONTINUE
```

```
      DO 375  R=1,NBPC(AUX2)
         DO 380  K=1,DIM
            PTF(K,R+NBPC(AUX1)) = PTDEB(CONV(AUX2,R),K)
380      CONTINUE
375   CONTINUE
C
C     ... on met dans NBPF le nombre de points contenus ...
C                    dans la classe obtenue
C
      NBPF = NBPC(AUX1)+NBPC(AUX2)
C
C     ... calcul de l'enveloppe convexe de la nouvelle classe ...
C
      WRITE(*,*)'RECOL. DES CLASSES ',AUX1,AUX2
C
      CALL CONVEXE(PTF,HYPERPLAN,VECTRA,FACE,POINT1,HYPER1,
     +                  FACE1,DIM,NBPF,NH,NF,VOLUME,DIMMAX,CONV)
C
C     ... mise a jour des volumes ...
C
      VOLU(AUX1) = VOLUME
      IF (AUX2.LT.NBC) THEN
         DO 385  I=AUX2,NBC-1
            VOLU(I) = VOLU(I+1)
385      CONTINUE
      ENDIF
C
C     ... mise a jour du tableau conv ...
C
      DO 386  I=1,NBPC(AUX2)
         CONV(AUX1,NBPC(AUX1)+I) = CONV(AUX2,I)
386   CONTINUE
C
      DO 387  I=AUX2,NBC-1
         DO 388  J=1,NBPC(I+1)
            CONV(I,J) = CONV(I+1,J)
388      CONTINUE
387   CONTINUE
C
C     ... mise a jour du nombre de points de la classe obtenue ...
C
      NBPC(AUX1) = NBPF
      IF (AUX2.LT.NBC) THEN
         DO 390  I=AUX2,NBC-1
            NBPC(I) = NBPC(I+1)
390      CONTINUE
      ENDIF
C
C     ... mise a jour du nombre de classes ...
C
      NBC = NBC-1
C
C     ... calcul de l'espace entre la nouvelle classe cree ...
C                    et les autres
C
      DO 395  I=1,NBC
         IF (I.NE.AUX1) THEN
C
C           ... stockage des coordonnees des points dans PTF ...
C
            DO 400  R=1,NBPC(I)
               DO 405  K=1,DIM
                  PTF(K,R) = PTDEB(CONV(I,R),K)
405            CONTINUE
400         CONTINUE
C
            DO 410  R=1,NBPC(AUX1)
               DO 415  K=1,DIM
                  PTF(K,R+NBPC(I)) = PTDEB(CONV(AUX1,R),K)
415            CONTINUE
410         CONTINUE
C
            NBPF = NBPC(I)+NBPC(AUX1)
C
C           ... calcul de l'enveloppe convexe de la classe ...
C               obtenue par la fusion de deux autres classes
```

```
C
C                 WRITE(*,*)'MISE A JOUR DE LA MAT DES ECARTS'
C
                  CALL CONVEXE(PTF,HYPERPLAN,VECTRA,FACE,POINT1,HYPER1,
     +                         FACE1,DIM,NBPF,NH,NF,VOLUME,DIMMAX,COND)
C
C                 ... calcul de l'ecart entre les deux classes considerees
C
                  IF (I.LT.AUX1) THEN
                     ECART (I,AUX1) = VOLUME - (VOLU(I)+VOLU(AUX1))
                  ELSE
                     ECART (AUX1,I) = VOLUME - (VOLU(I)+VOLU(AUX1))
                  ENDIF
C
               ENDIF
395         CONTINUE
C
C           ... mise a jour de la matrice des ecarts ...
C
            DO 420  I=AUX2,NBC
               DO 425  J=1,I-1
                  IF(J.NE.AUX1) THEN
                    IF (J.LT.AUX2) THEN
                       ECART(J,I) = ECART(J,I+1)

                    ELSE
                       ECART(J,I) = ECART(J+1,I+1)
                    ENDIF
                  ENDIF
425            CONTINUE
420         CONTINUE
C
            GOTO 350
C
         ENDIF
C
         critER = 0
         do i=1,nbrcla
            criter = criter+volu(i)
         enddo
C
         write(*,*)'critere:',criter
C        ------------------------------
C        ... affichage des resultats ...
C        ------------------------------
C
         WRITE(US,*)' APRES RECOLLEMENT '
C
         WRITE(US,*)' LE NOMBRE DE CLASSES EST :',NBRCLA
C
         DO 430  K=1,NBRCLA
C
            WRITE(US,*)'========================================'
            WRITE(US,*)
            WRITE(US,*)' VOLUME DE LA CLASSE NUMERO',K,' :',VOLU(K)
C
            WRITE(US,*)' LE NOMBRE DE POINTS EST : ',NBPC(K)
C
            WRITE(US,*)'LES POINTS SONT :'
C
            DO 435  I=1,NBPC(K)
               WRITE(US,*)(PTDEB(CONV(K,I),J),J=1,DIM)
435         CONTINUE
C
430      CONTINUE
C
         CLOSE(US)
         END
C
```

```
C=======================================================================
      DOUBLE PRECISION FUNCTION EQHYP(POINT,COEF,PT,DIM,DIMMAX)

      INTEGER PT,DIM,DIMMAX
      DOUBLE PRECISION POINT(4*DIMMAX,DIMMAX),COEF(DIMMAX+1)

      INTEGER I

      EQHYP = COEF(1)
      DO 5  I=1,DIM
         EQHYP = EQHYP + (COEF(I+1)*POINT(PT,I))
5     CONTINUE
      END
```

```
C=======================================================================

      SUBROUTINE DIVISIONDIM2 (M,SOMMET,DIMMAX,HYPT,NBH)

      BUT DE LA SOUS-ROUTINE :
      ************************

      Trouver les equations des hyperplans separateurs en dimension 2 d'un
      convexe dont les sommets sont donnes dans un ordre cyclique.

      TABLE DES PARAMETRES DE LA SOUS-ROUTINE :
      -----------------------------------------

      M : variable entiere d'ENTREE qui represente le nombre de sommets
          du convexe.

      DIMMAX : Constante entiere (ENTREE) qui contient la dimension maximale
               du probleme .

      NBH : variable entiere de SORTIE qui contient le nombre d'hyperplans
            separateurs.

      SOMMET : tableau d.p. (ENTREE) qui contient les coordonnees des M
               sommets.

      HYPT : tableau d.p. (SORTIE) de NBH lignes et 7 colonnes
             a chaque ligne ,on a :
                colonne 1 et 2 : les extremites du premier cote
                  " "   3 et 4 :  "     "       "    du second cote
                colonne 5 : le terme independant de l'hyperplan separateur.
                    "     6 : le coefficient de x  "      "          "

                colonne 7 :  "       "       " y

      DECLARATION  DES PARAMETRES :
      -----------------------------

      INTEGER M,DIMMAX,NBH

      DOUBLE PRECISION SOMMET(4*DIMMAX,*),HYPT(20,7)
```

TABLE DES VARIABLES LOCALES :

I : variable entiere qui represente le numero de l'hyperplan courant.

A,B : var entieres qui representent les extremites du premier cote.

C,D : var entieres qui representent les extremites du second cote.

STOP : var entiere qui represente quand la seconde boucle du programme s'arrete.

COEFH1,COEFH2 : var d.p. qui contiennent les trois coefficients des equations du premier (resp. du second) hyperplan.

EQHYP : var d.p. fonction declaree ci-dessus.

BOOL : tab. logique qui decrit comment se situe les points A,B,C,D par rapport a l'hyperplan considere.

GARDEH1,GARDEH2 : var logiques qui sont a 'true' si on garde le premier (resp le second) hyperplan. Ces deux variables sont toujours opposees.

DECLARATION DES VARIABLES LOCALES :

```
      INTEGER I,A,B,C,D,STOP
      DOUBLE PRECISION COEFH1(3),COEFH2(3),EQHYP
      LOGICAL BOOL(4),GARDEH1,GARDEH2
      EXTERNAL EQHYP

      I = 0

                        # # Pour chaque cote AB ##

      DO 10  A=1,M-1
         B = A+1
         IF (A.EQ.1) THEN
            STOP = M-1
         ELSE

            STOP = M
         ENDIF

                        # # Pour chaque cote CD non contigu a AB # #

         DO 15  C=A+2,STOP
            IF (C.LT.M) THEN
               D = C+1
            ELSE
               D = 1
            ENDIF
            I = I+1

              # # On calcule les coefficients du premier hyperplan # #

            COEFH1(2) = (SOMMET(A,2)-SOMMET(B,2))*
     +                  (SOMMET(D,2)-SOMMET(C,2))
            COEFH1(3) = (SOMMET(C,1)-SOMMET(D,1))*
     +                  (SOMMET(B,1)-SOMMET(A,1))
            COEFH1(1) = (SOMMET(A,1)*SOMMET(B,2))-
     +                  (SOMMET(B,1)*SOMMET(A,2))-
     +                  (SOMMET(C,1)*SOMMET(D,2))+
     +                  (SOMMET(D,1)*SOMMET(C,2))
```

```
C          # # On calcule les coefficients du second hyperplan # #
C
           COEFH2(2) = SOMMET(A,2)-SOMMET(B,2)-
     +                 SOMMET(C,2)+SOMMET(C,2)
           COEFH2(3) = -SOMMET(C,1)+SOMMET(D,1)+
     +                  SOMMET(B,1)-SOMMET(A,1)
           COEFH2(1) = (SOMMET(A,1)*SOMMET(B,2))-
     +                 (SOMMET(B,1)*SOMMET(A,2))+
     +                 (SOMMET(C,1)*SOMMET(D,2))-
     +                 (SOMMET(D,1)*SOMMET(C,2))
C
             BOOL(1) = (EQHYP(SOMMET,COEFH1,A,2,DIMMAX).LT.0)
             BOOL(2) = (EQHYP(SOMMET,COEFH1,B,2,DIMMAX).LT.0)
             BOOL(3) = (EQHYP(SOMMET,COEFH1,C,2,DIMMAX).LT.0)
             BOOL(4) = (EQHYP(SOMMET,COEFH1,D,2,DIMMAX).LT.0)
C
             * # Si le premier hyperplan coupe le convexe, on le garde # #
C
             GARDEH1=((BOOL(1).EQ.BOOL(2)).AND.(BOOL(3).EQ.BOOL(4))
     +                .AND.(BOOL(1).NE.BOOL(3)))
C
             BOOL(1) = (EQHYP(SOMMET,COEFH2,A,2,DIMMAX).LT.0)
             BOOL(2) = (EQHYP(SOMMET,COEFH2,B,2,DIMMAX).LT.0)
             BOOL(3) = (EQHYP(SOMMET,COEFH2,C,2,DIMMAX).LT.0)
             BOOL(4) = (EQHYP(SOMMET,COEFH2,D,2,DIMMAX).LT.0)
C
             * # Si le second hyperplan coupe le convexe, on le garde # #
C
             GARDEH2=((BOOL(1).EQ.BOOL(2)).AND.(BOOL(3).EQ.BOOL(4))
     +                .AND.(BOOL(1).NE.BOOL(3)))
C
C
             # # les resultats # #
C
                  HYPT(I,1) = A
                  HYPT(I,2) = B
C
                  HYPT(I,3) = C
                  HYPT(I,4) = D
                  IF (GARDEH1) THEN
                       WRITE(*,*)'NON REJET DE H1'
                     HYPT(I,5) = COEFH1(1)
                     HYPT(I,6) = COEFH1(2)
                     HYPT(I,7) = COEFH1(3)
                  ENDIF
                  IF (GARDEH2) THEN
                       WRITE(*,*)'NON REJET DE H2'
                     HYPT(I,5) = COEFH2(1)
                     HYPT(I,6) = COEFH2(2)
                     HYPT(I,7) = COEFH2(3)
                  ENDIF
                  NBH = I
C
15         CONTINUE
10       CONTINUE
         END
C
C
```

```
==========================================================================

SOUSROUTINE ORDRE(SOMEX,NB,DIM,DIMMAX,COTE,SOMMET,NSOM)
```

BUT DE LA SOUS-ROUTINE :
****

Ordonner les sommets d'un convexe de maniere cyclique .

TABLE DES PARAMETRES :

NB : Var entiere d'ENTREE qui represente le nombre de sommets au depart.

DIM : var entiere d'ENTREE qui contient la dimension du probleme.

DIMMAX : Constante entiere (ENTREE) qui contient la dimension maximale du probleme .

NSOM : var entiere de SORTIE qui contient le nombre de sommets du convex a la fin.

SOMEX : tab d.p. d'ENTREE qui contient les coordonnes des NB sommets de depart.

SOMMET : tab d.p. de SORTIE qui contient les coordonnes des NSOM sommets d'arrive.

DECLARATION DES PARAMETRES :

```
INTEGER NB,DIM,DIMMAX,NSOM
DOUBLE PRECISION SOMEX(4*DIMMAX,*),SOMMET(4*DIMMAX,*)
EXTERNAL COTE
```

TABLE DES VARIABLES LOCALES :

IND : vecteur qui contient les indices des sommets qui restent a ordonner. Si l'indice = 0, le sommet est deja ordonne.

I,J,K,L : variables de boucle

P : indice du sommet courant.

PTSUR : vecteur qui donne les indices des sommets qui sont alignes.

Z : nombre de sommets alignes hormis I et IPREC.

Y : nombre de sommets que l'on retire a chaque iteration .

IPREC : valeur de I a la precedente iteration .

A,B : indices des deux points les plus eloignes sur une meme droite.

COORD : idem

S1,S2 : vecteurs qui contiennent les coordonnes des sommets courants.

DIST : la distance entre les point A et B

DISTMAX : maximum de DIST.

FORMULE : calcul intermediaire qui permet de voir si S1,S2 et le premier point ordonne sont alignes.

REJET : variable logique qui est a "TRUE" si tous les sommets sont d'un meme cote de la droite formee par S1 et S2.

CHANGES2 : variable logique qui est a "TRUE" s'il faut changer la valeur de S2.

FIN : variable logique qui est a "TRUE" si FORMULE est nul et donc tous les sommets sont ordonnes cycliquement.

```
C
C     DECLARATION DES VARIABLES LOCALES :
C     -----------------------------------
C
      INTEGER IND(8),J,P,I,K,PTSUR(6),L,Q,Z,Y,IPREC,A,B,COORD(2)
      DOUBLE PRECISION S1(2),S2(2),DIST,DISTMAX,FORMULE
      LOGICAL REJET,CHANGES2,FIN
C
C                    # #  Initialisation # #
C
      IPREC = 1
      Y = 0
      NSOM = NB
      IND(1) = 0
      DO 5  K=2,8
         IND(K)=K
5     CONTINUE
C
      DO 10  J=1,DIM
         S1(J) = SOMEX(1,J)
10    CONTINUE
C
      P = 1
C
      DO 11 J=1,2
         SOMMET(P,J) = S1(J)
11    CONTINUE
C
15    IF (P.LT.NSOM) THEN
C
C           # # tant que tous les sommets n'ont pas ete traites * #
C
         I = 1
         REJET = .TRUE.
C
C           # # tant que l'on rejette le sommet S2 et que l'on n'a pas # #
C                considere tous les sommets faire ...
C
20       IF ((REJET.EQ..TRUE.).AND.(I.LE.NB)) THEN
            I = I+1
25          IF (IND(I).EQ.0) THEN
               I = I+1
               GOTO 25
            ENDIF
            DO 26 J=1,2
               S2(J) = SOMEX(I,J)
26          CONTINUE
C
            IF (P.GE.3) THEN
C
C              # # apres avoir ordonne au moins 3 sommets,voir si les # #
C                   2 sommets S1,S2 que l'on considere sont alignes avec
C                   le premier ordonne.
C
               FORMULE = (S2(1)-S1(1))*(SOMEX(1,2)-S1(2))
     +                   -(S2(2)-S1(2))*(SOMEX(1,1)-S1(1))
               FIN = (FORMULE.EQ.0)
               IF (FIN) THEN
C
C                # # si oui, alors la boucle est bouclee. # #
C                     c'est la fin, il faut mettre a jour NSOM
C
                  Z = 0
                  DO 17 K=1,NB
                     IF (IND(K).NE.0) THEN
                        Z = Z+1
                     ENDIF
17                CONTINUE
                  NSOM = NSOM - Z
                  GOTO 100
C
               ENDIF
            ENDIF
```

```
      # # si non, on continue # #
      CALL COTE(S1,S2,SOMEX,NB,DIMMAX,REJET,PTSUR)
      CHANGES2 = .FALSE.
      # # si tous les sommets sont du meme cote que AB # #
      IF (.NOT.REJET) THEN
      L = 1
      Z = 0
      # # tant qu'il y a des points des points alignes # #
      IF ((PTSUR(L).NE.0).AND.(L.LE.6)) THEN
      # # si ce n'est pas un des 2 sommets consideres, # #
      mettre son indice a zero et compter combien de
      sommets sont dans le meme cas.
      IF ((PTSUR(L).NE.I).AND.(PTSUR(L).NE.IPREC)) THEN
      IND(PTSUR(L)) = 0
      Z = Z + 1
      ENDIF
      L = L+1
      GOTO 27
      ENDIF
      Y = Y + Z
      ENDIF
      IF ((Z.GE.1).AND.(.NOT.REJET)) THEN
      # # Choisir les points les plus eloignes pour # #
      qu'ils deviennent les sommets
      DISTMAX = 0
      DO 28 A=1,5
      DO 29 B=A+1,6
      IF ((PTSUR(A).NE.0).AND.(PTSUR(B).NE.0)) THEN
      DIST = (SOMEX(PTSUR(A),1)-SOMEX(PTSUR(B),1))** 2
      + (SOMEX(PTSUR(A),2)-SOMEX(PTSUR(B),2))** 2
      IF (DIST.GT.DISTMAX) THEN
      DISTMAX = DIST
      COORD(1) = A
      COORD(2) = B
      ENDIF
      ENDIF
      CONTINUE
      CONTINUE
      # # Mettre a jour les coordonnes de S2 # #
      A = COORD(1)
      B = COORD(2)
      IF (PTSUR(A).EQ.IPREC) THEN
      IF (PTSUR(B).NE.I) THEN
      S2(1) = SOMEX(PTSUR(B),1)
      S2(2) = SOMEX(PTSUR(B),2)
      CHANGES2 = .TRUE.
      ENDIF
      ELSE
      IF (PTSUR(A).NE.I) THEN
      S2(1) = SOMEX(PTSUR(A),1)
      S2(2) = SOMEX(PTSUR(A),2)
      CHANGES2 = .TRUE.
      ENDIF
      ENDIF
      ENDIF
      GOTO 20
      ENDIF
```

```
C
C            # # Mettre a jour l'indice IPREC # #
C
          IF (CHANGES2) THEN
             if (ptsur(b).ne.iprec) then
             IPREC = PTSUR(B)
             else
             iprec = ptsur(a)
             endif
C
          ELSE
             IPREC = I
          ENDIF
          IND(I) = 0
          P = P+1
C
C            # # mettre a jour SOMMET,S1 et NSOM # #
C
          DO 30  J=1,2
             SOMMET(P,J) = S2(J)
             S1(J) = S2(J)
30        CONTINUE
          NSOM = NSOM - Y
          Y = 0
          GOTO 15
       ENDIF
100    CONTINUE
       END
C
C
C      ==================================================================
C
C
       SUBROUTINE COTE(A,B,C,NB,DIMMAX,REJET,PTSUR)



       BUT DE LA SOUS-ROUTINE :
       ************************

       Voir si tous les points (au nombre de NB) dont les coordonnees sont
       dans le tableau C sont d' un meme cote de la droite AB.

C   Si non, REJET est mis a "TRUE".
C   PTSUR est un tableau ou l'on trouve les indices des points qui sont
C   sur la droite AB.
C
C   TABLE DES PARAMETRES :
C   ----------------------
C
C   A,B : contiennent les coordonnees du point A et B  de la droite
C         consideree.
C         (ENTREE)
C
C   NB : nombre de points du tableau C.(ENTREE)
C
C   C : tableau qui contient les coordonnees des points que l'on veut situer
C       par rapport a la droite AB.(ENTREE)
C
C   DIMMAX : Dimension maximale du probleme.(ENTREE)
C
C   REJET : variable logique qui est a "TRUE" si tous les points de C
C           sont d'un meme cote de la droite AB et a "FALSE" sinon.
C           (SORTIE)
C
C   PTSUR : vecteur qui contient les indices des points de C qui sont
C           situe sur la droite AB.
C           (SORTIE)
C
C   DECLARATION DES PARAMETRES :
C   ----------------------------
C
C
C   INTEGER NB,DIMMAX,PTSUR(6)
C
C   DOUBLE PRECISION A(2),B(2),C(4*DIMMAX,*)
C
C   LOGICAL REJET
C
```

TABLE DES VARIABLES LOCALES :

I,J,K : variable de boucles.

EQ : Evaluation de l'expression analytique de la droite AB en un point du tableau C.

BOOL : variable logique qui situe le premier point de C par rapport a la droite AB.

BOOLPT : vecteur logique qui situe les autres points de C par rapport a la droite AB.

DECLARATION DES VARIABLES LOCALES :

```
      INTEGER I,J,K
C
      DOUBLE PRECISION EQ
C
      LOGICAL BOOL,BOOLPT(8)
C
C
      DO 4 I=1,6
         PTSUR(I) = 0
4     CONTINUE
C
      I = 1
      EQ = ((B(1)-A(1))*(C(I,2)-A(2)))-((B(2)-A(2))*(C(I,1)-A(1)))
      BOOL = (EQ.LE.0)
      REJET = BOOL
      I = 2
      J = 1
5     IF (I.LE.NB) THEN
         EQ = ((B(1)-A(1))*(C(I,2)-A(2)))-((B(2)-A(2))*(C(I,1)-A(1)))
         IF (EQ.EQ.0) THEN
            PTSUR(J) = I
            J = J + 1
         ENDIF
         BOOLPT(I) = (EQ.LE.0)
         I = I+1
         GOTO 5
      ENDIF
      K = 2
10    IF ((BOOLPT(K).EQ.BOOL).AND.(K.LE.NB)) THEN
         K= K + 1
         GOTO 10
      ENDIF
      IF (K.EQ.NB+1) THEN
         REJET = .FALSE.
      ELSE
         REJET = .TRUE.
      ENDIF
      END
C
C     ================================================================
```

Références

[1] ANDERBERG M.R. ,
Cluster analysis for applications.
Academic Press (1973), New-York.

[2] CELEUX G. et co-auteurs ,
Classification automatique des données.
Edition Bordas (1989), Paris.

[3] CHALON A. ,
Détermination du nombre de classes en classification automatique :
comparaison de quelques méthodes et applications.
Mémoire (1990), Facultés Universitaires Notre-Dame de la Paix, Namur.

[4] DAWAGNE J.M. et DELPEREE F. ,
Une simplification des méthodes basées sur le critère de Rasson
en classification automatique et en analyse discriminante, applications.
Mémoire (1988), Facultés Universitaires Notre-Dame de la Paix, Namur.

[5] GENDARME V. ,
Implémentation d'un nouvel algorithme basé sur la technique de maximisation du vide,
sur le critère des hypervolumes et sur la détermination du nombre de classes
en classification automatique.
Mémoire (1991), Facultés Universitaires Notre-Dame de la Paix, Namur.

[6] HAMMERSLEY J.M. ,
Stochastic models for the distributions of particules in space.
Suppl. Adv. Applied Probability 47-68 (1972).

[7] HARDY A. ,
Statistique et classification automatique : un modèle,
un nouveau critère, des algorithmes, des applications.
Thèse de doctorat (1983), Facultés Universitaires Notre-Dame de la Paix, Namur.

[8] HENRY M. ,
Implémentation d'un nouvel algorithme en classification automatique.
Description, exemples, application.
Mémoire (1990), Facultés Universitaires Notre-Dame de la Paix, Namur.

[9] KARR A.F. ,
Point processes and their statistical inference.
Second edition. Marcel Dekker,inc.(1991),New-York.

[10] MEUNIER P.H. et RASSON J.P. ,
Une nouvelles méthode de classification basée sur le critère de Rasson.
Département de mathématique, F.N.D.P. (1982) Namur

[11] RIPLEY B.D. ,
Stochastic simulation.
John Wiley & sons (1987),New-york.

[12] WAERENBURGH L. ,
Méthode exacte en analyse discriminante basée sur le critère de Rasson.
Application à la télédetection.
Mémoire (1990), Facultés Universitaires Notre-Dame de la Paix, Namur.

Table des Matières