MASTER EN SCIENCES MATHÉMATIQUES

Contribution spécifique du critère des hypervolumes à la détermination du nombre de classes dans les espaces multidimensionnels

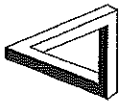
Pirard, Florence

Award date:
2000

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Contribution spécifique du critère des hypervolumes à la détermination du nombre de classes dans des espaces multidimensionnels



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

Promoteur : A. Hardy

Florence PIRARD

Année Académique 1999-2000

En premier lieu, je voudrais adresser mes plus vifs remerciements à Monsieur André Hardy pour avoir accepté de diriger ce mémoire. Sa disponibilité et ses conseils ont été une aide précieuse.

Je tiens également à remercier Laurence Dumortier et Vincent Bertholet pour l'aide qu'ils m'ont apportée dans le cadre du volet informatique de ce travail.

Je ne veux pas oublier Fabian Bastin pour son générateur de nombres aléatoires.

Enfin, je tiens aussi à remercier mes parents, Anne-Françoise, Bernard, Fabian, Michel et Sébastien pour m'avoir soutenue au cours de ces quatres années d'études.

Résumé

La classification est utilisée dans le but de résoudre le problème suivant : comment diviser une population donnée d'objets ou d'individus décrite par un ensemble de caractéristiques en un nombre relativement restreint de sous-groupes d'objets ou d'individus semblables. Un des problèmes liés à cette discipline est de déterminer le nombre de classes présentes dans les données. Plusieurs travaux relatifs à ce sujet ont été réalisés. Un de ceux-ci fut effectué par Milligan et Cooper. Dans ce travail, ils ont comparé trente méthodes de détermination du nombre de classes, ce qui leur permit de les classer. Deux autres travaux ont été réalisés afin de comparer les six premières méthodes du classement de Milligan et Cooper avec celles basées sur le critère des hypervolumes développées aux FUNDP. Cette comparaison ayant été essentiellement dressée pour des ensembles de données où le nombre de caractéristiques observées est de deux, nous l'avons poursuivie pour des ensembles de données où ce nombre varie entre 4 et 8. Les résultats montrent que pour les ensembles de données utilisés, les méthodes basées sur le critère des hypervolumes donnent généralement de meilleurs résultats.

Abstract

Classification techniques are used to partition a set of objects described by several features into smaller subsets or clusters. The members of each cluster share some common features. The determination of the true number of clusters constitutes one of the classification problems. This problems has been studied by many researchers, like Milligan and Cooper, who have compared and ranked 30 procedures used to determine the number of clusters. Two other studies on 2 dimensional data sets, have been carried out to compare the top 6 performing methods identified by Milligan and Cooper with those based on the hypervolumes criterion developped in the FUNDP. In this work, we have extended these 2 latter studies, considering data sets whose dimensions range from 4 to 8. As a result, the methods based on the hypervolumes criterion have been generally found to perform better for the data sets under study.

Table des matières

Errata

p 30 : TSS est la trace de R.

p 34 : Pour calculer $\min(\Gamma)$, les valeurs des d_{ij} doivent être ordonnées de la plus petite à la plus grande et celles des $C(x_i,x_j)$ de la plus grande à la plus petite. Le minimum est obtenu en prenant la moitié de la somme des produits des d_{ij} par les $C(x_i,x_j)$ appartenant au même rang.

Chapitre 1

Introduction

Classer des individus ou des objets est une activité inhérente à l'esprit humain. Afin de synthétiser une réalité complexe, l'homme a tendance à établir des catégories qui regroupent des individus ou des objets ayant des caractéristiques communes. Ainsi, si on lui soumet la liste suivante : voiture, moto, camion, vélo, tracteur, monture, il groupera directement les véhicules avec moteur et ceux sans, ou encore il distinguera les différents groupes suivant le nombre de roues. Ceci est un exemple de classification.

De nombreuses disciplines employent la classification. Que ce soit en biologie, en zoologie, en botanique, en géologie, en agronomie ou encore en médecine, pour n'en citer que quelques-unes, la classification est utilisée pour résoudre le problème suivant : *Comment diviser une population donnée d'objets ou d'individus, décrite par un ensemble de caractéristiques, en* ***un nombre*** *relativement restreint de sous-groupes d'objets ou d'individus semblables.* Ainsi par exemple, en biologie végétale, une étude fut effectuée sur 78 populations de Campanula rotundifolia L., S.L. provenant de toute l'Europe et caractérisées par 28 attributs tels que la hauteur moyenne des plantes, le nombre moyen de fleurs par tige, la couleur du pollen, la forme de la corolle. L'objectif de cette recherche était de mettre en évidence des gradients morphologiques entre différentes espèces ([5]).

Différentes méthodes de classification existent pour résoudre de tels problèmes. Elles permettent d'obtenir une partition des données en un certain nombre de groupes ; en général, celui-ci doit être fourni par l'utilisateur. Mais quand les données proviennent de la réalité, ce dernier ne connaît pas toujours, a priori, le nombre de groupes correspondant à la partition la plus naturelle. Pour pallier à cet inconvénient, de nombreuses techniques ont été proposées. Celles-ci permettent de déterminer le nombre de classes présentes dans les données et se basent sur les résultats obtenus par les méthodes de classification pour différents

nombres de classes. C'est le cas, par exemple, des méthodes de détermination du nombre de classes basées sur le critère des hypervolumes et développées aux FUNDP par A. Hardy.

Plusieurs travaux comparant des méthodes de détermination du nombre de classes ont été réalisés. Un de ceux-ci fut effectué par Milligan et Cooper. Dans leur étude, ces derniers ont appliqué une trentaine de ces méthodes à divers ensembles de données; ceci leur a permis d'établir un classement fondé sur la régularité des méthodes à retrouver le nombre correct de groupes. Ce travail en inspira d'autres et notamment un, comparant les six meilleures méthodes issues du classement de Milligan et Cooper avec les méthodes basées sur le critère des hypervolumes. Celui-ci ayant été réalisé essentiellement pour des ensembles de données en dimension 2, nous allons le poursuivre, à travers ce mémoire, en appliquant ces méthodes à des ensembles de données de dimension supérieure.

Dans le chapitre suivant, nous commencerons par rappeler les grandes étapes d'une classification. Ensuite, nous détaillerons dans le chapitre 3, les travaux relatifs à la détermination du nombre de classes sur lesquels nous nous sommes appuyés, les méthodes basées sur le critère des hypervolumes et celles issues du classement de Milligan et Cooper. Puis, nous passerons à l'application de ces méthodes. Nous présenterons tout d'abord les programmes informatiques qui ont été réalisés ainsi que les résultats trouvés en dimensions 2 et 3. Enfin, nous exposerons les résultats que nous avons obtenus pour des ensembles de données de dimension 4 à 8.

Chapitre 2

La classification automatique

2.1 Les étapes d'une classification automatique

En général, un problème de classification peut être décrit par cinq étapes :

1° *La collecte des données*

L'utilisateur définit les individus ou les objets qu'il désire étudier ainsi que les différentes variables qui les caractériseront. Leur choix influencera fortement les résultats de l'analyse.

2° *Le calcul des proximités*

L'utilisateur choisit une mesure de similarité ou de dissimilarité entre les paires d'individus ou d'objets. Le choix de cet indice est une première étape pour synthétiser l'information contenue dans les données.

3° *La constitution des groupes*

L'utilisateur fait le choix d'une méthode de classification et d'un algorithme.

4° *L'interprétation des résultats*

L'utilisateur identifie les différents groupes formés suivant leur nombre, leurs formes, leurs éléments...

5° *La validation des résultats*

L'utilisateur détermine la qualité de la classification.

Remarquons qu'un problème de classification ne comporte pas toujours ces cinq étapes. Il se peut que l'utilisateur dispose directement de la matrice des similarités au lieu de la matrice de données. De plus, certaines méthodes de classification définissent des critères qui ne se basent pas sur une mesure de similarité ou de dissimilarité mais sur le calcul d'une autre valeur ; c'est le cas pour la méthode des hypervolumes que nous présenterons à la fin de ce chapitre.

Poursuivons ce chapitre en précisant les trois premières étapes d'une classification automatique.

2.2 La collecte des données

Les individus doivent être choisis de façon à obtenir des informations pertinentes sur la population considérée. En général, cette dernière est trop importante et ne peut pas être étudiée dans sa totalité ; c'est pourquoi, dans ce cas, un échantillon représentatif de la population est sélectionné.

Ensuite, les variables doivent être choisies de manière à caractériser au mieux les individus dans le contexte étudié. C'est probablement leur choix qui aura le plus d'influence sur la formation des groupes.

Une fois les individus et les variables déterminés, l'utilisateur pourra recueillir les données. Supposons que la population soit constituée de n individus : $x_1, x_2, \ldots, x_n$ et que sur chacun d'eux p variables $(v_1, v_2, \ldots, v_p)$ soient observées. Les données se représentent par une matrice $(n \times p)$, appelée **matrice des données**. Elle est de la forme :

$$M = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}.$$

Notons que dans la suite de ce travail, nous supposerons que les variables utilisées sont quantitatives, ainsi chaque individu pourra être représenté par un

point dans un espace euclidien de dimension p.

2.3 Le calcul des proximités

Grâce à la matrice des données, l'utilisateur possède une représentation du phénomène qu'il désire étudier. Malheureusement, ces données brutes ne permettent pas de découvrir la structure d'un simple coup d'oeil. C'est pourquoi, il est intéressant de définir une mesure de proximité permettant de déterminer le degré de similarité des paires d'objets. Cette mesure est appelée **indice de similarité ou de dissimilarité.**

Par la suite, nous utiliserons comme mesure de proximité la distance euclidienne non pondérée :

$$d_{ij}^2 = \sum_{k=1}^{p} (x_{ik} - x_{jk})^2.$$

Notons que lorsque le nombre de variables observées est petit ($p \leq 3$), cette mesure de similarité entre les objets peut être visualisée grâce à la représentation de ceux-ci dans un espace euclidien de dimension p. Il s'agit de la longueur du segment entre les différents points.

Si on calcule cet indice pour chaque paire d'objets, on obtient la **matrice des proximités** :

$$D = \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1p} \\ d_{21} & d_{22} & \cdots & d_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{np} \end{pmatrix}.$$

Cette matrice est symétrique et les éléments sur la diagonale principale sont nuls.

2.4 La constitution des groupes

Dans cette partie consacrée à la constitution des groupes, nous allons voir comment il est possible de synthétiser l'information contenue dans la matrice des proximités en remplaçant des distances entre objets par des distances entre groupes d'objets. Nous commencerons par préciser le problème de classification.

2.4.1 Le problème de classification

Soit E l'ensemble des n individus à classer sur lesquels on a observé p variables :

$$E = \{x_1, \ldots, x_n\}.$$

On recherche une partition de E en k classes (k fixé) :

$$P = \{C_1, \ldots, C_k\}.$$

Soit Pk l'ensemble des partitions de E en k classes. A chaque partition P de Pk, on peut associer un critère de classification :

$$\begin{array}{rccc} W: & Pk & \to & R^+ \\ & P & \rightsquigarrow & W(P,k) \end{array}$$

Pour trouver la partition optimale en k classes, il suffit de minimiser ou de maximiser le critère. Donc, la partition optimale est :

$$P^* = \{C_1^*, \ldots, C_k^*\} \text{ telle que } W(P^*,k) = \min_{P \in Pk} W(P,k),$$
$$\text{ou} = \max_{P \in Pk} W(P,k).$$

Les différentes méthodes de classification se distinguent par le critère utilisé.

Comme l'ensemble des données est fini, on pourrait penser résoudre le problème de classification en énumérant toutes les partitions et on choisirait ensuite la meilleure selon un critère de classification. Pratiquement, ceci est impensable. En effet, le nombre de partitions de n objets en k classes est le nombre de Stirling du deuxième ordre ([5]) :

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

avec $S(n,k) = k\,S(n-1,k) + S(n-1,k-1)$,
et $S(1,1) = 1$,
$S(1,n) = 0 \quad (\text{où } n \neq 1\ )$.

Par exemple :

S(59,2)	$\approx$	10^{18},
S(15,3)	=	2 375 101,
S(20,4)	=	45 232 115 901,
S(100,5)	$\approx$	10^{68}.

Si un ordinateur évalue un million de partitions par seconde, il lui faudrait 8 jours pour déterminer la partition optimale de 20 individus en 5 classes et plus de 2 444 siècles si le nombre d'individus passe à 30. De plus, lorsque le nombre de classes est inconnu, le nombre total de partitions est la somme des nombres de Stirling. Celui-ci est appelé nombre de Bell ([5]) :

$$B(n) = \sum_{k=1}^{n} S(n,k) \; = \; e^{-1} \sum_{k=1}^{\infty} \frac{k^n}{k!}.$$

Par exemple, pour $n = 10$, $B(n)$ vaut 115 975.

Heureusement, l'utilisation des méthodes de classification permet de réduire considérablement le nombre de calculs à effectuer.

2.4.2 Les méthodes de classification

Les méthodes de classification sont nombreuses. Nous allons essayer, dans cette partie, de situer celles que nous utiliserons par la suite. Parmi toutes les méthodes, on distingue dans un premier temps les *méthodes monothétiques* et les *méthodes polythétiques*. Les premières ont pour objectif de rechercher une partition construite à partir de la matrice des données. Celle-ci s'obtient par une suite de divisions en deux classes, en ne tenant compte que d'une seule variable à la fois. Les méthodes polythéthiques, quant à elles, tiennent compte simultanément de toutes les variables et c'est la matrice des proximités qui est utilisée. Elles se différencient suivant la structure souhaitée de la classification : partition, recouvrement, hiérarchie de partitions ... On distingue donc les *méthodes non hiérarchiques* et les *méthodes hiérarchiques*. En ce qui concerne les premières, nous ne les détaillerons pas ici ; citons juste la méthode des hypervolumes que nous présenterons à la fin de ce chapitre. Les secondes, quant à elles, permettent de générer une famille de partitions telle que les groupements ou les divisions successifs des individus forment une hiérarchie. Les méthodes hiérarchiques se divisent en deux groupes : les *méthodes agglomératives* et les *méthodes divisives*.

1° Les méthodes agglomératives

Elles partent de n singletons et réunissent à chaque étape les deux classes les plus semblables, selon le critère de similarité choisi. Le processus s'arrête lorsque tous les individus sont rassemblés dans une seule classe.

2° Les méthodes divisives

Ces méthodes partent d'une classe contenant les n individus et divisent en deux, à chaque étape, la classe contenant les groupes les plus dissemblables selon le critère choisi. Le processus s'arrête quand on obtient la partition en n singletons.

Rappelons que c'est le choix du critère de classification qui différencie les méthodes. Avant de présenter les méthodes hiérarchiques que nous avons utilisées, faisons quelques remarques à leur sujet. Tout d'abord, notons que les hiérarchies obtenues en utilisant une méthode agglomérative et divisive peuvent être différentes. De plus, la hiérarchie obtenue peut dépendre de l'ordre de lecture des données. Remarquons également qu'en général, il est préférable d'utiliser les méthodes agglomératives ; en effet, ces dernières requièrent souvent moins de calculs

pour le passage d'une itération à la suivante que les méthodes divisives. C'est la raison pour laquelle, nous utiliserons par la suite, des méthodes hiérarchiques agglomératives. Signalons, qu'un gros inconvénient de ces méthodes provient du fait que l'affectation d'un individu à un groupe à une étape n'est jamais remise en cause par la suite.

2.4.3 Présentation des méthodes hiérarchiques utilisées

La méthode du saut minimum ([10])

Cette méthode porte également le nom de *méthode du voisin le plus proche.* La distance entre deux classes C_i et C_j d'une partition est la plus petite distance séparant un point d'une classe avec un point de l'autre :

$$d(C_i, C_j) = \min_{\substack{x \in C_i \\ y \in C_j}} d(x, y).$$

De façon schématique :

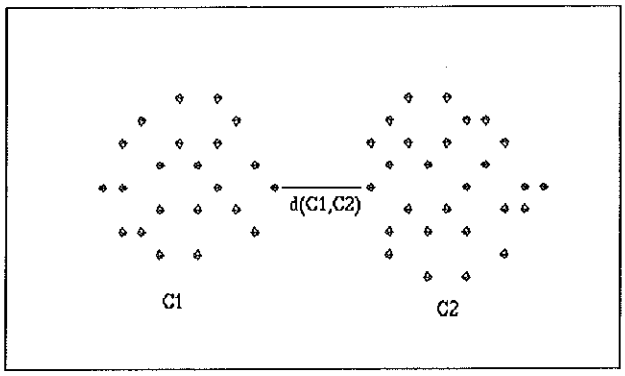


FIG. 2.1 – La distance du saut minimum

A chaque étape, on fusionne donc les deux classes qui sont les plus proches au sens du critère.

Un inconvénient de cette méthode est que la hiérarchie peut dépendre de l'ordre de lecture des données. Si la plus petite distance entre deux groupes est mesurée entre plusieurs paires, elle regroupera les deux premiers. Cette méthode

est également peu robuste car si on perturbe légèrement les données, on peut obtenir une partition totalement différente. Cette méthode a l'avantage de détecter les classes bien séparées et les classes allongées (*effet de chaînage*). Toutefois, cet effet peut être ennuyeux lorsque les classes sont reliées par des objets formant des ponts entre elles. Dans ce cas, les classes obtenues par la méthode du saut minimum ne reflètent pas la structure véritable des données.

La méthode du saut maximum ([10])

Cette méthode est également appelée *méthode du voisin le plus éloigné.* Elle consiste à mesurer la distance entre deux groupes par la plus grande distance qui les sépare :

$$d(C_i, C_j) = \max_{\substack{x \in C_i \\ y \in C_j}} d(x, y).$$

De façon schématique :

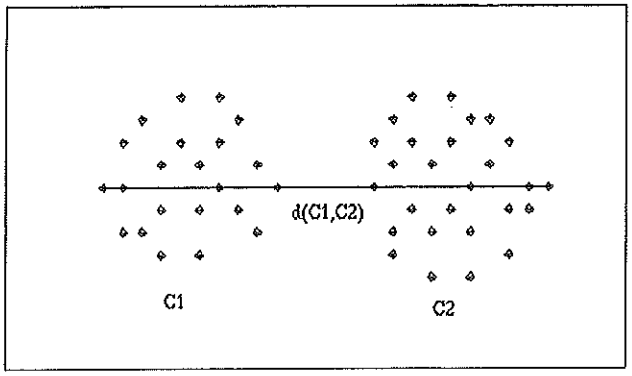


FIG. 2.2 – La distance du saut maximum

Tout comme la méthode du saut minimum, celle du saut maximum est peu robuste et la hiérarchie obtenue dépend de l'ordre de lecture des données. Elle détecte également les classes bien séparées, mais fonctionne mal quand les classes sont allongées. Elle a tendance à former des groupes hypersphériques.

La méthode de la moyenne ([10])

Cette méthode mesure la distance entre deux classes C_i et C_j comprenant respectivement n_i et n_j éléments, par la valeur moyenne des distances inter-classes :

$$d(C_i, C_j) = \sum_{\substack{x \in C_i \\ y \in C_j}} \frac{d(x,y)}{(n_i\, n_j)}.$$

Remarquons que le critère de la moyenne ne fait pas intervenir des distances intra-classes. Il garantit donc que deux classes fusionnées soient composées d'individus en moyenne proches mais ne garantit pas que les classes soient homogènes. Notons également que les résultats obtenus en utilisant le critère de la moyenne diffèrent peu de ceux obtenus par la méthode du voisin le plus éloigné. Cette méthode a donc tendance à former des groupes hypersphériques.

La méthode de Ward ([5])

La distance entre deux classes d'objets C_i et C_j peut être mesurée par la différence entre le moment centré d'ordre 2 des classes fusionnées et le moment centré de chacune des deux classes :

$$d^2(C_i, C_j) = M^2(C_i \cup C_j) - M^2(C_i) - M^2(C_j),$$

où

$$\begin{aligned} M^2(C_i) &= \sum_{k=1}^{p} \sum_{i \in C_i} (x_{ik} - \bar{x_k})^2 \\ &= \sum_{i,j \in C_i} \frac{d_{ij}}{n_i}. \end{aligned}$$

Le moment centré d'ordre 2 d'une classe d'objets est donc égal à la somme du carré des écarts des objets au centre de gravité de la classe. Il est aussi égal à la moyenne des distances euclidiennes entre toutes les paires d'objets de la classe.

Ce critère se ramène [Jambu 1972] à une pondération de la distance entre les centres de gravité :

$$d(C_i, C_j) = \left[\frac{n_i\, n_j}{n_i + n_j}\right]^{\frac{1}{2}} \parallel g(C_i) - g(C_j) \parallel,$$

où $\parallel . \parallel$ est la norme euclidienne usuelle,
et $g(C_i)$ et $g(C_j)$ sont les centres de gravité des classes C_i et C_j.

A chaque itération, on fusionne les deux groupes qui conduisent à un accroissement minimum du critère des moindres carrés. Cette méthode tend également à former des groupes hypersphériques.

2.4.4 La méthode des hypervolumes ([12])

Contrairement aux méthodes de classification que nous venons de décrire, la méthode des hypervolumes repose sur un modèle statistique de la classification. L'approche est basée sur la théorie des processus ponctuels et de l'estimation par la technique du maximum de vraisemblance d'un domaine convexe compact.

Le modèle

Le modèle se base sur les hypothèses suivantes :

1° Les points sont distribués dans k sous-domaines disjoints $D_1, D_2, \ldots, D_k$.

2° Les variables aléatoires comptant le nombre de points dans des régions disjointes sont indépendantes.

3° Le nombre moyen de points dans chaque région est proportionnel à la mesure de Lebesgue de cette région.

Un seul processus satisfait à ces trois conditions, il s'agit du **processus de Poisson stationnaire**. Dans un tel processus, la variable indiquant le nombre de points dans chaque région a une distribution de Poisson dont le paramètre est

la mesure de Lebesgue de cette région.

Nous traitons donc un problème de classification lorsque les points observés sont engendrés par un processus de Poisson et sont distribués dans k domaines disjoints $(D_i)_{1 \leq i \leq k}$ que nous souhaitons retrouver.

Solution statistique

Considérons un ensemble de points engendré par un processus de Poisson dans k domaines disjoints $(D_i)_{1 \leq i \leq k}$. Le problème soulevé par la classification, est donc l'estimation des frontières des k ensembles $D_1, D_2, \ldots, D_k$.

Soit $x = (x_1, x_2, \cdots, x_n)$ la réalisation du processus dans D, l'union des domaines disjoints D_i. Nous noterons par $C_i \subset (x_1, x_2, \cdots, x_n)$ l'ensemble des observations appartenant à D_i $(1 \leq i \leq k)$. La fonction de vraisemblance s'écrit sous la forme :

$$f_D(x) = \left(\frac{1}{m(D)}\right)^n \prod_{i=1}^{n} I_D(x_i),$$

où $m(D)$ est la mesure de Lebesgue de D
et $I_D(x_i)$ est la fonction indicatrice de D.

En appliquant la méthode du maximum de vraisemblance, on obtient que le domaine D pour lequel la vraisemblance est maximale, est parmi ceux qui contiennent tous les points, celui dont la mesure de Lebesgue est minimale.

Mais si nous n'imposons pas de contraintes sur la structure de D, nous constatons qu'il existe beaucoup de solutions triviales à ce problème. Une contrainte suffisante est d'exiger que les classes soient **convexes** ([18]).

Par conséquent, le maximum de la fonction de vraisemblance sera atteint par la partition pour laquelle **la somme des mesures de Lebesgue des enve-**

loppes convexes des k sous-domaines est minimale.

Le problème de classification et le critère

Nous désirons trouver une partition de l'ensemble E des individus en k classes disjointes $C_1, C_2, \ldots, C_k$.

Le critère de classification est le suivant : **la somme des mesures de Lebesgue des enveloppes convexes des k classes.** De façon mathématique :

$$\begin{array}{rccl} W : & Pk & \rightarrow & R^+ \\ & P = \{C_1, C_2, \ldots, C_k\} & \rightsquigarrow & W(P,k) = \sum_{i=1}^{k} m(H(C_i)) \end{array}$$

où Pk est l'ensemble des partitions possibles de l'ensemble des données en k classes,
et $m(H(C_i))$ est la mesure de Lebesgue de l'enveloppe convexe des points appartenant à C_i.

Le problème est donc de trouver $P^* \in Pk$ tel que :

$$\begin{aligned} W(P^*,k) &= \min_{P \in P_k} W(P,k) \\ &= \min \sum_{i=1}^{k} m(H(C_i)). \end{aligned}$$

Par exemple dans R^2, on recherche les k groupes de points tels que la somme des aires de leurs enveloppes convexes soit minimale. De façon plus générale, dans R^p, on essaye de trouver les k groupes de points tels que la somme des hypervolumes de leurs enveloppes convexes soit minimale.

Chapitre 3

La détermination du nombre de classes

3.1 Introduction

Quand le nombre de variables observées est de deux ou de trois, c'est-à-dire quand les données peuvent se représenter graphiquement, le classement des individus ou des objets ne présente aucune difficulté. L'utilisateur, grâce à son propre jugement, peut discerner les différents groupes.

Dans les espaces de dimension supérieure, on ne peut plus procéder de façon visuelle. L'utilisation des méthodes de classification constitue une première étape du traitement des données. Mais, une fois les résultats de ces méthodes obtenus, il reste encore à les analyser. En effet, les quatre méthodes hiérarchiques que nous avons présentées précédemment, fournissent un classement de l'ensemble des individus ou des objets pour tous les nombres de classes, mais elles ne permettent pas de répondre aux questions suivantes : Quel est le nombre de classes présentes dans les données et quelle est la structure réelle des données ?

Afin d'y répondre, des méthodes permettant de déterminer le nombre de classes ou de vérifier l'absence de structure ont été proposées. Ce sont elles que nous étudierons dans ce chapitre et que nous appliquerons à divers ensembles de données dans les chapitres suivants. Mais avant, nous préciserons la notion de classe et présenterons quelques travaux relatifs à la détermination du nombre de classes.

3.2 La notion de classe naturelle

Jusqu'à présent, nous avons fait allusion à plusieurs reprises à la notion de classe mais nous ne l'avons pas encore définie ; c'est ce que nous allons essayer de faire dans cette partie.

Si on se réfère au problème de classification, nous pouvons caractériser une classe comme étant composée d'objets semblables, c'est-à-dire d'objets ayant des caractéristiques communes ou proches. Ce sont ces dernières qui permettent d'identifier la classe et de la distinguer des autres.

Dans le chapitre précédent, nous avons supposé que chaque objet, caractérisé par p variables, pouvait être représenté par un point dans un espace de dimension p. Nous avons également vu que dans ce cas, la similarité entre une paire d'objets peut être mesurée par l'intermédiaire de la distance les séparant. Dès lors, nous pouvons déclarer que les classes sont des groupes de points dans un espace de dimension p, tels que la distance entre deux points d'une même classe est plus petite que la distance entre deux points appartenant à des classes différentes. Mais ceci n'est pas toujours vrai. Pour s'en convaincre, considérons l'exemple suivant :

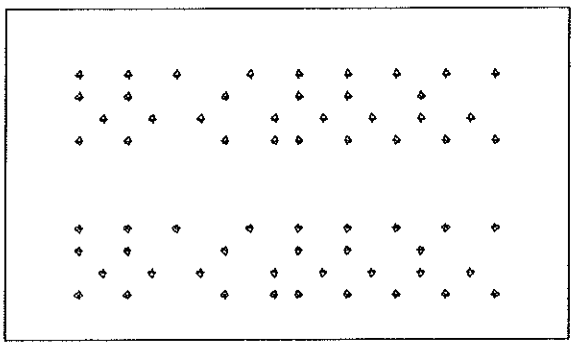


FIG. 3.1 – Deux classes parallèles

Nous distinguons clairement deux classes. Pourtant, ces dernières ne satisfont pas la définition. En effet, des paires de points appartenant à des classes différentes sont plus proches que des paires appartenant à la même classe.

Pour éviter ce genre de problème, nous adopterons une définition plus intuitive. Supposons que les individus ou les objets que l'on désire étudier soient représentés par un point dans un espace de dimension p, nous pouvons alors définir une classe comme étant une région de l'espace contenant une densité relativement grande de points ; les différentes classes étant séparées par des régions contenant relativement peu de points. Dans les espaces de dimension 2 ou 3, la détection de ces amas de points peut se faire visuellement. Ces classes sont qualifiées de *naturelles.*

La qualité d'une classification sera donc jugée suivant qu'elle retrouve les classes naturelles ou pas. Par exemple, comparons les résultats obtenus en utilisant la méthode du saut minimum et du saut maximum pour les données représentées par la figure 3.1. Les partitions en deux classes obtenues en utilisant la méthode du saut minimum et celle du saut maximum sont réprésentées respectivement par les figures 3.2 et 3.3.

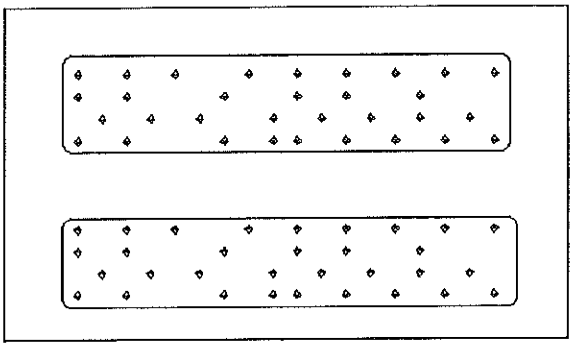


FIG. 3.2 – Classification obtenue par la méthode du saut minimum

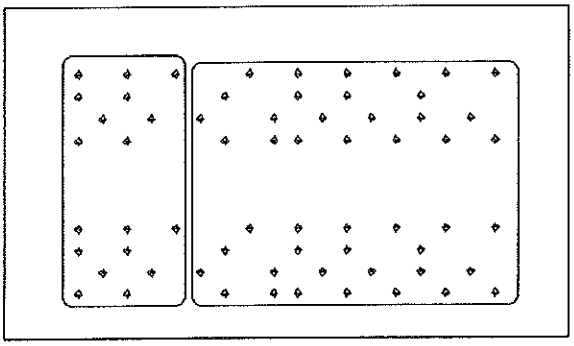


FIG. 3.3 – Classification obtenue par la méthode du saut maximum

Nous constatons donc que la méthode du saut minimum retrouve les classes attendues tandis que celle du saut maximum nous donne d'autres classes. Pour notre exemple, la méthode du saut minimum peut donc être qualifiée de meilleure puisqu'elle fournit les classes naturelles.

Remarquons que nous pouvions nous attendre à un tel résultat. En effet, nous avons vu que la méthode du saut maximum favorisait la formation de classes hypershériques, or ici, les classes naturelles sont allongées. Cet exemple permet donc de soulever un problème beaucoup plus général. Certaines méthodes de classification imposent une structure particulière aux classes, au lieu de trouver leur véritable structure.

Notons, également qu'en général, les jeux de données que nous utiliserons pour comparer les différentes méthodes de détermination du nombre de classes, ont des classes naturelles évidentes qui, contrairement aux données réelles, sont toujours connues préalablement. Nous pourrons donc facilement juger de la qualité des méthodes de classification et de détermination du nombre de classes.

3.3 Les travaux déjà réalisés

Différents travaux ayant pour but de comparer des méthodes de détermination du nombre de classes ont déjà été réalisés. Parmi ceux-ci, citons celui effectué par Milligan et Cooper ([15]), par Gordon ([11]) ou encore le mémoire de P. André ([1]) et celui de J.-F. Deschamps ([7]). Nous allons examiner successivement ces quatre travaux.

3.3.1 Le travail de Milligan et Cooper ([15])

Dans leur travail, Milligan et Cooper ont comparé trente méthodes existantes de détermination du nombre de classes. Cette comparaison leur a permis d'établir un classement de ces dernières sur base de leur régularité ; les premières étant celles qui donnaient le plus souvent les résultats attendus.

Pour établir ce classement, Milligan et Cooper ont créé divers ensembles de données générés de façon aléatoire. Chacun de ces ensembles contenait 50 points dans un espace de dimension 4, 6 ou 8 et comportait 2, 3, 4 ou 5 classes. De

plus, ces dernières présentaient la caractéristique d'être bien séparées les unes des autres. Au total, Milligan et Cooper disposaient de 108 ensembles de données. En vue d'obtenir une classification pour appliquer les méthodes de détermination du nombre de classes, ils utilisèrent quatre méthodes hiérarchiques qui sont celles que nous avons présentées précédemment.

Dans leur article, Milligan et Cooper insistent sur le fait que leurs ensembles de données présentaient des classes bien séparées. Le classement pourrait donc être bouleversé pour des ensembles de données présentant une structure plus particulière. Toutefois, il est difficile de croire qu'une méthode donnant de mauvais résultats pour des structures nettes, fonctionne mieux dans des cas plus complexes.

Parmi toutes les règles de détermination du nombre de classes qui existent, Milligan et Cooper ont opté pour des règles indépendantes des méthodes de classification. Ils ont également éliminé celles qui laissaient place à la subjectivité humaine comme par exemple les méthodes graphiques. Seules les méthodes où la règle de décision était automatique ont été conservées. De plus, chaque méthode a été utilisée dans de bonnes conditions, c'est-à-dire que quand il y avait un doute sur le nombre de classes, la décision a toujours été prise en faveur de la méthode.

Pour chacune des méthodes, Milligan et Cooper ont synthétisé leurs résulats sous forme de tableaux du genre de celui-ci :

Calinski and Harabasz	2	3	4	5	overall
2 or fewer	-	-	1	0	1
1 too few	-	12	6	0	18
Correct level	96	95	97	102	390
1 too many	3	0	3	6	12
2 too many	4	0	1	0	5
3 or more	5	1	0	0	6

TAB. 3.1 – Résultats obtenus par Milligan et Cooper pour la méthode de Calinski et Harabasz

Ce tableau répertorie le nombre de fois que la règle, ici celle de Calinski et Harabasz, a retrouvé le nombre correct de classes, ce nombre étant indiqué au-dessus des colonnes. Il indique également le nombre de fois où elle a trouvé trop ou trop peu de classes. Notons que Milligan et Cooper jugeaient que trouver un

nombre de classes inférieur au nombre réel était plus grave que le contraire car de l'information est perdue lors de la fusion de deux classes.

Dans l'exemple, nous constatons que la méthode de Calinski et Harabasz a retrouvé, pour les ensembles de données contenant 3 classes, 95 fois le nombre correct, 12 fois, elle a donné une classe en moins et 1 fois elle a fourni trois classes ou plus de trop.

En plus du classement établi, Milligan et Cooper ont constaté que de nombreuses méthodes présentaient leur plus mauvais résultat quand le nombre de classes était de deux.

Par la suite, nous étudierons et utiliserons les six premières méthodes du classement de Milligan et Cooper; il s'agit de la méthode de Calinski et Harabasz, la méthode de Duda et Hart, la méthode du C-index, la méthode Gamma, la méthode de Beale et la méthode du Cubic Clustering Criterion présente dans le logiciel statistique SAS.

3.3.2 Le travail de Gordon ([11])

L'objectif du travail de Gordon était de comparer les cinq premières méthodes issues du classement de Milligan et Cooper sur des ensembles de points présentant une structure différente de ceux utilisés par ces derniers.

Ce travail se base sur le fait que spécifier une seule valeur pour le nombre de classes peut dans certain cas mener à une représentation trompeuse de la structure de l'ensemble de données; en effet, ces dernières peuvent parfois être expliquées par différents nombres de classes. C'est pourquoi, Gordon compara la capacité des cinq premières méthodes de Milligan et Cooper à déterminer les différents nombres de classes adéquats pour la structure des données. Pour effectuer cette comparaison, Gordon utilisa des ensembles de données où le nombre de classes pouvait être de trois ou de douze. De plus amples renseignements sur ce travail sont disponibles dans l'article de Gordon([11]).

3.3.3 Le travail de P. André ([1])

L'objectif du travail de P. André était de comparer les six meilleures méthodes de détermination du nombre de classes issues du classement de Milligan et Cooper avec les méthodes basées sur le critère des hypervolumes. Cette comparaison s'est effectuée sur des ensembles de données de dimension 2, beaucoup moins nombreux que ceux de Milligan et Cooper mais présentant des structures plus particulières : classes allongées, effectifs fortement différents... Ce travail a permis de mettre en évidence certaines caractéristiques des méthodes de détermination du nombre de classes utilisées. Les conclusions de ce travail seront présentées dans le chapitre suivant.

3.3.4 Le travail de J.-F. Deschamps ([7])

Ce travail poursuit celui effectué par P. André. Il comporte deux parties. La première avait pour objectif d'une part d'appliquer les six meilleures méthodes de détermination du nombre de classes de Milligan et Cooper aux résultats fournis par la méthode de classification des hypervolumes et d'autre part d'appliquer les trois méthodes de détermination du nombre de classes basées sur le critère des hypervolumes aux hiérarchies de partitions fournies par les quatre méthodes de classification présentées auparavant. Ceci a été réalisé dans des espaces de dimension 2. La seconde partie consistait en l'application des six meilleures méthodes issues du classement de Milligan et Cooper aux quatre méthodes de classification hiérarchiques dans des espaces de dimension 3. L'objectif de cette deuxième partie était de voir si les conclusions de P. André se confirmaient et d'essayer de préciser les biais de ces méthodes. Les ensembles de données en dimension 3 sont une généralisation de ceux utilisés en dimension 2. Les principaux résultats seront également repris dans le chapitre suivant.

3.3.5 Notre travail

L'objectif de ce mémoire est de poursuivre le travail de P. André et de J.-F. Deschamps. En effet, nous souhaitons comparer, dans des espaces de dimension supérieure à 3, deux des trois méthodes de détermination du nombre de classes basées sur le critère des hypervolumes, avec les six meilleures méthodes du classement de Milligan et Cooper. En vue d'effectuer cette comparaison, nous avons créé différents ensembles de données dont certains sont une généralisation de ceux utilisés en dimension 2 et 3. Nous espérons ainsi déterminer si les conclusions de P. André et J.-F. Deschamps se confirment dans des espaces de dimension supé-

rieure à 3.

3.4 Les méthodes de détermination du nombre de classes basées sur le critère des hypervolumes

3.4.1 La méthode du coude ([13])

Cette méthode de détermination du nombre de classes est graphique. Elle consiste à tracer le graphe de la valeur du critère de classification en fonction du nombre de classes. La présence d'un coude sur le graphique permet d'évaluer le nombre de classes présentes dans les données.

Pour le critère des hypervolumes, on trace donc la courbe de la somme des mesures de Lebesgue des enveloppes convexes des classes en fonction du nombre de groupements. Ensuite, il suffit de rechercher le coude.

Exemple

Appliquons cette méthode à un ensemble de données ayant deux classes bien séparées. On distingue bien sur le graphe le coude pour k=2.

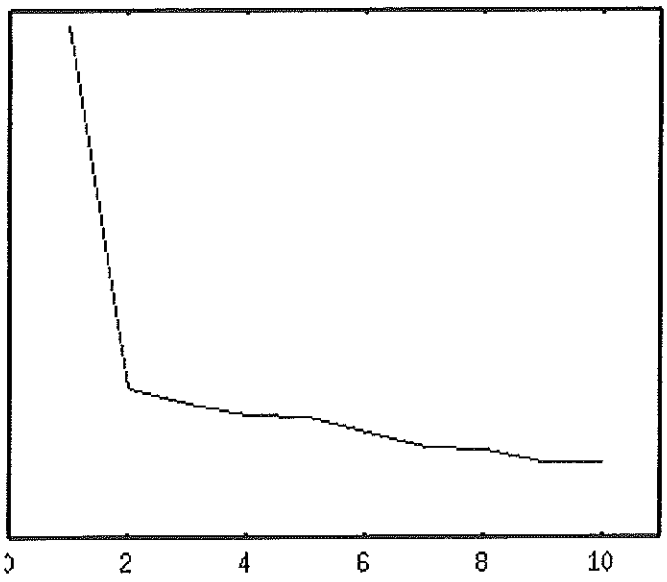


FIG. 3.4 - Valeur du critère en fonction du nombre de classes

La présence de ce coude est liée au fait que quand le nombre de classes augmente, la valeur du critère de classification décroît. En effet lors de la division d'une classe en deux, on constate une diminution de la valeur du critère, mais lors de la division d'une classe naturelle, l'hypervolume enlevé est minime et dans ce cas, la valeur du critère se stabilise.

3.4.2 Une méthode basée sur un test du quotient de vraisemblance ([13])

La méthode des hypervolumes, comme nous l'avons déjà signalé, présente l'avantage de reposer sur un modèle statistique. Celui-ci permit à A. Hardy d'établir un test du quotient de vraisemblance capable de déterminer le nombre optimal de classes.

Soit $X_1, X_2, \ldots, X_n$ la réalisation d'un processus de Poisson homogène dans t domaines convexes disjoints d'un espace euclidien à m dimensions. Nous souhaitons tester si une subdivision en k classes est significativement meilleure qu'une subdivision en $k-1$ classes. Les hypothèses du test peuvent donc se formuler de la façon suivante :

$$\begin{aligned} H_0 &: t = k, \\ H_1 &: t = k - 1. \end{aligned}$$

Notons 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.

Le quotient de vraisemblance est de la forme :

$$
\begin{aligned}
Q(x) &= \frac{\frac{1}{(m(H(C)))^n}}{\frac{1}{(m(H(D)))^n}} \\
&= \left(\frac{W(P,k)}{W(P,k-1)}\right)^n \\
&= S^n.
\end{aligned}
$$

La région de rejet est de la forme :

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

La distribution de la statistique S reste malheureusement inconnue. Nous ne pouvons donc pas étudier les propriétés théoriques du test, ni déterminer le niveau de signification avec lequel on accepte ou rejette l'hypothèse H_0. Toutefois, S présente une propriété intéressante : S est compris entre 0 et 1.

En pratique, nous utiliserons la règle de décision suivante : rejeter H_0 si S prend de trop grandes valeurs, c'est-à-dire si S est proche de 1. Donc, si k_0 est la première valeur pour laquelle on rejette H_0, on acceptera qu'il y ait $k_0 - 1$ classes naturelles présentes dans les données.

Remarquons que cette règle s'explique : lors de la division d'une classe naturelle en deux, l'hypervolume enlevé est petit, dès lors la valeur de $W(P,k)$ est proche de la valeur de $W(P,k-1)$. Le quotient est alors proche de 1.

3.4.3 Une méthode basée sur l'estimation d'un ensemble convexe ([13])

Signalons que dans nos applications, nous n'utiliserons pas cette règle. Nous la présentons dans le but d'être complet.

Cette méthode se base sur la solution du problème proposé par Kendall et qui est le suivant : étant donné la réalisation d'un processus de Poisson homogène d'intensité inconnue dans un ensemble convexe compact D, trouver D. ([18])

La solution à ce problème fut proposée par Rasson et Ripley ([18]) et Rasson ([17]). L'estimation de D consiste en une dilatation de l'enveloppe convexe $H(x)$ des n observations $x = (x_1, \ldots, x_n)$ à partir de son centre de gravité :

$$D' = g(H(x)) + c\, s(H(x)),$$

où $g(H(x))$ est le centre de gravité de $H(x)$,
$s(H(x)) = H(x) - g(H(x))$,
c est une constante de dilatation qui peut être estimée par $\sqrt{\frac{n}{n-v_n}}$ (approximation de Moore ([16])) avec v_n le nombre de sommets de $H(x)$.

Une méthode de détermination du nombre de classes se base sur ce résultat.

Remarquons tout d'abord que la réalisation d'un processus de Poisson dans l'union C de k sous-domaines disjoints $C_1, C_2, \ldots, C_k$ peut être considérée comme la réalisation d'un processus de Poisson de même intensité dans k domaines $C_1, C_2, \ldots, C_k$ (Neveu 1974).

Notons par $C_i'^k$ l'estimation de la classe C_i^k $(i = 1, \ldots, k)$ d'une partition en k classes. La procédure permettant de déterminer le nombre de classes est basée sur le fait que les domaines doivent être disjoints. Elle prend la forme suivante :

- Si $\forall\{i,j\} \subset \{1,\ldots,t\}\ i \neq j\ C_i'^t \bigcap C_j'^t = \emptyset$,
si $\forall t' \in N\backslash\{0,1\}\ t' < t$ et $\forall\{i,j\} \subset \{1,2,\ldots,t'\}$, $C_i'^{t'} \bigcap C_j'^{t'} = \emptyset$,
alors, on conclut que la partition naturelle contient au moins t classes et on examine la partition optimale en $t+1$ classes.

- Si $\exists\{i,j\} \subset \{1,2,\ldots,t\}\ i \neq j\ C_i'^t \bigcap C_j'^t \neq \emptyset$,
si $\forall t' \in N\backslash\{0,1\}\ t' < t$ et $\forall\{i,j\} \subset \{1,2,\ldots,t'\}\ C_i'^{t'} \bigcap C_j'^{t'} = \emptyset$,
alors, on conclut que l'ensemble des données contient $t-1$ classes naturelles.

- Si $C_1'^2 \bigcap C_2'^2 \neq \emptyset$,
alors, on conclut à l'absence de structure dans l'ensemble de données.

3.5 Présentation des six méthodes issues du classement de Milligan et Cooper ([15])

3.5.1 La méthode de Calinski et Harabasz ([4])

Supposons que la population ou l'échantillon que nous étudions, soit constitué de n individus sur lesquels p variables ont été observées. De plus, supposons également que ceux-ci soient classés dans k groupes.

Nous noterons :

- d_{ij} la distance euclidienne entre les individus i et j.

Soit R la matrice de dispersion totale; on peut décomposer R comme étant :

$$R = B + W$$

où B est la matrice de dispersion inter-groupe,
et W est la matrice de dispersion intra-groupe.

Les éléments de ces différentes matrices sont de la forme :

$$r_{jl} = \sum_{i=1}^{n} (x_{ij} - \overline{x}_j)(x_{il} - \overline{x}_l),$$

$$w_{jl} = \sum_{c=1}^{k} \sum_{i=1}^{n_c} (x_{ij} - \overline{x}_{jc})(x_{il} - \overline{x}_{lc}),$$

$$b_{jl} = \sum_{c=1}^{k} n_c (\overline{x}_{jc} - \overline{x}_c)(\overline{x}_{lc} - \overline{x}_c),$$

avec

- k : le nombre de groupes,
- n_c : l'effectif du groupe c,
- n : l'effectif de la population,
- p : le nombre de variables,
- j, l : indices de variables avec $1 \leq j \leq p$ et $1 \leq l \leq p$,
- $\overline{x}_j$: la moyenne générale de la variable j,
- $\overline{x}_{jc}$: la moyenne générale de la variable j dans le groupe c.

Nous utiliserons également comme notation :

- $BGSS$ (Between Group Sum of Squares) $\equiv$ trace B,
- $WGSS$ (Within Group Sum of Squares) $\equiv$ trace W,
- TSS (Total Sum of Squares) $\equiv$ trace T.

On obtient comme relation :

$$-\text{Trace}\, W = WGSS = \text{Trace}\, R_1 + \text{Trace}\, R_2 + \ldots + \text{Trace} R_k,$$

$$\text{où Trace}\, R_g = n_g^{-1} \left(d^2_{12\,(g)} + d^2_{13\,(g)} + \ldots + d^2_{n_{g-1}n_g\,(g)}\right);$$

$$-\text{Trace}\, R = TSS = \frac{1}{n} \left(d^2_{12} + d^2_{13} + \ldots + d^2_{n-1\,n}\right).$$

La méthode de Calinski et Harabasz consiste à calculer à chaque étape le critère du rapport de variance (VRC) :

$$VRC = \frac{\frac{BGSS}{k-1}}{\frac{WGSS}{n-k}}.$$

Cet indice possède de nombreuses propriétés. Celles-ci sont disponibles dans l'article original ([4]) ou dans le mémoire de P. André ([1]).

Pour déterminer le nombre de classes présentes dans les données, Calinski et Harabasz suggèrent de calculer VRC pour $k = 2, 3, \ldots$ et de choisir le nombre k pour lequel VRC a un maximum absolu ou relatif ou au moins une croissance plus rapide. Si les valeurs de VRC ont une croissance monotone avec celles de k, la meilleure partition des points est celle où chaque point forme un groupe.

3.5.2 La méthode de Duda et Hart ([8])

Soit χ un ensemble de n individus $x_1, \ldots, x_n$ sur lequel d variables sont observées. Nous noterons par χ_i l'ensemble des points du i-ème groupe et par m_i la moyenne de ces points.

Définissons :

$$J_e(k) = \sum_{i=1}^{k} \sum_{x \in \chi_i} \|x - m_i\|^2$$

$J_e(k)$ est la somme des carrés des erreurs qui surviennent si on représente les n individus par les k centres des classes.

Remarquons que $J_e(k)$ diminue de façon monotone quand k augmente car à chaque division d'un groupe en deux sous-groupes, la somme des carrés des erreurs diminue. Si l'ensemble de données présente k_0 groupes bien séparés, on peut s'attendre à ce que $J_e(k)$ diminue rapidement jusque $k = k_0$ et puis diminue

beaucoup plus lentement pour atteindre 0 quand $k = n$.

Duda et Hart ont formulé un test permettant de décider si oui ou non l'ensemble χ des n individus forme plus qu'un groupe et ceci grâce à la somme des écarts à la moyenne.

Leur méthode se base sur l'hypothèse suivante :

> H_0 : les points viennent d'une population normale de moyenne μ et de matrice de covariance $\sigma^2 I$.

Duda et Hart propose de rejeter H_0 au niveau p si :

$$\frac{J_e(2)}{J_e(1)} < 1 - \frac{2}{\pi d} - \alpha \sqrt{\frac{2\left(1 - \frac{8}{\pi^2 d}\right)}{n d}}$$

où α est déterminé par :

$$p = 100 \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}u^2} \, du.$$

Cette méthode de détermination du nombre de classes peut être appliquée avec un algorithme agglomératif; il suffit de calculer à chaque étape :

$$\frac{-\frac{J_e(2)}{J_e(1)} + 1 - \frac{2}{\pi d}}{\sqrt{\frac{2\left(1 - \frac{8}{\pi^2 d}\right)}{n d}}}.$$

On utilise la règle suivante : dès que cette expression dépasse la valeur choisie pour α, on rejette l'hypothèse H_0 qui correspond à l'existence d'un seul groupe; c'est que la fusion des deux groupes n'était plus justifiée. Donc, si la valeur de l'indice est plus grande que celle de α pour $k = k_0$, on conclut qu'il y a $k_0 + 1$ classes dans les données.

Plusieurs valeurs de α ont été proposées. Dans leur étude, Milligan et Cooper ont utilisé un α valant 3.20. Gordon quant à lui a suggéré de prendre $\alpha = 4$. En fait, ils ont chacun fait choix du α qui leur donnait les meilleurs résultats sur les exemples qu'ils ont considérés.

3.5.3 La méthode du "C-index" ([14])

Supposons que la population étudiée se compose de n individus $(x_1, \ldots, x_n)$ répartis dans k groupes.

Définissons :

$$C(x_i, x_j) = \begin{cases} 1 & \text{si } x_i \text{ et } x_j \text{ sont dans la même classe } (i \neq j), \\ 0 & \text{sinon,} \end{cases}$$

et

$$\begin{aligned} \Gamma &= \frac{1}{2} \sum_{j=1}^{n} \sum_{i=1}^{n} d_{ij}\, C(x_i, x_j) \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} d_{ij}\, C(x_i, x_j) \end{aligned}$$

où d_{ij} est la distance euclidienne entre les objets i et j.

Nous avons donc que :

$$\Gamma = \sum_{i=1}^{k} \text{des distances entre les objets du i-ème groupe.}$$

Pour déterminer le nombre de classes, nous allons utiliser une normalisation de Γ. Celle-ci permet de définir un indice appelé C-index qui est de la forme :

$$\text{C-index} = \frac{\Gamma - \min(\Gamma)}{\max(\Gamma) - \min(\Gamma)}.$$

Le minimum et le maximum sont obtenus en ordonnant d'une part les valeurs des d_{ij} et d'autre part les valeurs des $C(x_i, x_j)$. Pour le minimum, il suffit de les disposer de la plus grande à la plus petite et ensuite de prendre la moitié de la somme des produits des d_{ij} par les $C(x_i, x_j)$ appartenant au même rang. Pour obtenir le maximum, on procédera de la même façon si ce n'est que les valeurs des d_{ij} et celles des $C(x_i, x_j)$ sont ordonnées cette fois de la plus petite à la plus grande.

C'est la valeur minimale de cet indice qui est utilisée pour indiquer le nombre de classes.

3.5.4 La méthode Gamma ([2])

Soit un ensemble de n objets $\{o_1, o_2, \ldots, o_n\}$. Supposons que ces objets soient classés dans k groupes.

Notons par d_{ij} la distance euclidienne entre o_i et o_j.

Définissons :

$$T_l(o_i, o_j) = \begin{cases} 0 & \text{si } o_i \text{ et } o_j \text{ sont dans la même classe,} \\ 1 & \text{sinon.} \end{cases}$$

De plus, si o_i et o_j appartiennent au même groupe, définissons également :

$$n_l(o_i, o_j) = \sharp \{ \{o_r, o_t\} : T_l(o_r, o_t) = 1 \text{ et } d(o_r, o_t) < d(o_i, o_j) \}.$$

En fait, il s'agit du nombre de couples d'objets n'appartenant pas à la même classe et qui sont plus proches que o_i et o_j.

Nous pouvons alors définir l'indice α_l :

$$\alpha_l = \frac{\sum_{i<j} n_l(o_i, o_j)}{\max \sum_{i<j} n_l(o_i, o_j)},$$

où le maximum est pris sur toutes les partitions possibles en k classes en gardant le même nombre d'objets par classe,
et les sommes sont effectuées sur les paires d'objets $\{o_i, o_j\}$ qui appartiennent à la même classe.

Remarquons que le numérateur de l'indice α_l représente la somme sur toutes les paires d'objets appartenant à la même classe, du nombre d'objets appartenant à des classes différentes et étant plus proches que les paires d'objets considérés dans la somme. L'indice α_l représente donc la proportion de paires d'objets qui sont dans un "mauvais groupe".

On peut alors définir l'indice γ :

$$\gamma = 1 - 2\alpha_l.$$

Cet indice varie entre -1 et 1. Ceci vient du fait que α_l est compris entre 0 et 1. Si tous les objets sont classés correctement α_l vaut 0 et donc γ prend la valeur 1. Dans le cas contraire, si tous les objets sont mal placés, γ vaut -1.

Pratiquemment, nous rechercherons la valeur maximale de l'indice, celle-ci devant être la plus proche possible de 1.

3.5.5 La méthode de Beale ([3])

Initialement, Beale a proposé un test permettant de juger si une division en c_2 classes était meilleure qu'une division en un plus petit nombre de classes c_1. Ce test fut adapté pour les méthodes hiérarchiques ; il permet, tout comme celui de Duda et Hart, de décider si la fusion de deux groupes est justifiée ou pas.

Soit n individus sur lesquels p variables ont été mesurées.

Notons par :

- W_1 la somme des carrés des distances à l'intérieur d'un ensemble de points,
- W_2 la somme des carrés des distances à l'intérieur des deux groupes obtenus en divisant l'ensemble initial en deux.

Le test consiste à calculer la statistique :

$$\frac{\frac{W_1 - W_2}{W_2}}{\left(\frac{n-1}{n-2}\right)^{2^{\frac{2}{p}}} - 1}$$

et à la comparer avec une distribution de Fisher-Snedecor à p degrés de liberté au numérateur et $(n-2)p$ degrés de liberté au dénominateur.

La règle d'arrêt est la suivante : si k_0 est la première valeur de l'indice qui entraîne le rejet de l'hypothèse correspondant à la fusion de deux classes, le nombre correct de classes est $k_0 + 1$.

Pour les données en dimension 4 que nous traiterons dans le chapitre suivant, nous comparerons donc la valeur de l'indice avec une loi $F_{p,(n-2)p}$, dont la valeur est de 2.37 pour un niveau de précision de 0.05 et de 3.32 pour un niveau de précision égal à 0.01.

3.5.6 La méthode CCC ([19])

Le critère de classification cubique est le test statistique de détermination du nombre de classes présent dans le logiciel SAS. A chaque étape, l'indice calculé est :

$$CCC = \ln\left(\frac{1 - E(R)^2}{1 - R^2}\right) \frac{\sqrt{\frac{np^*}{2}}}{(0.001 + E(R)^2)^{1.2}}$$

où

- R^2 est la proportion de variance expliquée par les classes,
- $E(R^2)$ est déterminé sous l'hypothèse que les données ont été générées par une distribution uniforme,
- n est le nombre d'observations,
- p^* est une estimation de la dimensionalité déterminée par une analyse en composantes principales.

C'est la valeur de l'indice correspondant à un pic maximal qui indique le nombre de classes présentes dans les données. Si cette valeur est supérieure à 2, la classification obtenue est bonne; si elle est comprise entre 0 et 2, on a des classes possibles. Des valeurs négatives indiquent des points isolés.

Chapitre 4

Comparaison des six meilleures méthodes de détermination du nombre de classes issues du classement de Milligan et Cooper avec celles basées sur le critère des hypervolumes

4.1 Introduction

Le but de ce chapitre est d'appliquer les huit méthodes de détermination du nombre de classes présentées au chapitre précédent, aux résultats fournis par quatre méthodes de classification hiérarchiques pour divers ensembles de données et ceci en vue d'effectuer une comparaison des méthodes de détermination du nombre de classes.

Mais avant, rappelons les méthodes hiérarchiques et les méthodes de détermination du nombre de classes que nous avons utilisées. Pour les premières, il s'agit de :

- la méthode du saut minimum,
- la méthode du saut maximum,
- la méthode de la moyenne,
- la méthode de Ward.

Les huit méthodes de détermination du nombre de classes sont les suivantes :

- les méthodes basées sur le critère des hypervolumes :

  - la méthode du coude (M1),
  - le test du quotient de vraisemblance (M2),

- les méthodes issues du classement de Milligan et Cooper :

  - la méthode de Calinski et Harabasz (m1),
  - la méthode de Duda et Hart (m2),
  - la méthode du C-index (m3),
  - la méthode Gamma (m4),
  - la méthode de Beale (m5),
  - la méthode CCC (m6).

Les ensembles de données sur lesquels nous avons appliqué ces méthodes ont une structure particulière. Ils sont au nombre de huit, il s'agit de :

- données avec trois classes bien séparées,
- données avec deux classes bien séparées,
- données avec deux classes parallèles,
- données avec trois classes parallèles,
- données "gros-petit" éloignés,
- données "gros-petit" proches,
- données sans structure,
- données avec un pont entre les classes.

Signalons encore que chacun de ces ensembles a été généré aléatoirement dans des espaces de dimension 4 à 8. Afin que le lecteur se rende bien compte de la structure de ces données, nous accompagnerons les résultats obtenus en dimension 4 de figures représentant les projections des données dans le plan des deux premiers facteurs d'une analyse en composantes principales.

Avant d'exposer ces résultats, nous allons présenter les programmes que nous avons utilisés pour réaliser cette étude ainsi que les principales conclusions du travail de P. André et de J.-F. Deschamps.

4.2 Présentation des programmes

4.2.1 Programme pour les méthodes basées sur le critère des hypervolumes

Tout d'abord, signalons qu'un programme calculant l'enveloppe convexe d'un ensemble de points ainsi que son hypervolume dans des espaces multidimensionnels nous a été fourni. Ce dernier nous a permis d'écrire facilement un programme calculant la valeur du critère des hypervolumes pour une partition obtenue en utilisant une des quatre méthodes de classification.

Supposons que nous désirions calculer la valeur du critère des hypervolumes pour un ensemble de données partitionné en k classes. Signalons que pour obtenir cette partition suivant la méthode hiérarchique souhaitée, nous avons utilisé le logiciel SAS ([20]). Notre programme (procédure crithyp) travaille sur le fichier contenant ce résultat de la façon suivante :

- Tout d'abord, pour chaque classe, il calcule son enveloppe convexe ainsi que l'hypervolume de cette dernière.

- Ensuite, il additionne les k hypervolumes. Cette somme est la valeur du critère.

Pour obtenir la valeur du test du quotient de vraisemblance, il suffit d'appliquer cette procédure pour un nombre de classes égal à k et k-1 et ensuite de faire le quotient de ces deux valeurs (procédure testhyp).

Signalons que de plus amples renseignements au sujet de ces deux procédures sont disponibles dans le code du programme (voir syllabus annexe).

4.2.2 Programme pour les méthodes issues du classement de Milligan et Cooper

Le progamme initial a été écrit par A.D. Gordon (voir syllabus annexe). Pour chacune des quatre méthodes hiérarchiques, il recherche la hiérarchie de partitions correspondante en procédant de façon agglomérative. De plus, à chaque étape, il calcule la valeur de l'indice des cinq premières méthodes de détermination du nombre de classes de Milligan et Cooper.

Pour chacune des quatre méthodes de classification, on obtient un tableau du genre de celui-ci :

saut minimum	Gamma	Duda et Hart	Beale	Calinski et Harabasz	C-index
k=n-1					
k=n-2					
⋮					
k=2					
k=1					

TAB. 4.1 – Tableau fourni par le programme de Gordon

où n est le nombre d'individus dans l'ensemble de données.

Ensuite, une fois que l'on possède ces tableaux, afin de déterminer le nombre de classes, il suffit d'appliquer les règles de décision pour les différentes méthodes.

Remarques

Le sixième indice du classement de Milligan et Cooper est le Cubic Clustering Criterion, il est obtenu en utilisant la procédure *cluster* de SAS. La façon dont il est calculé dans cette dernière est identique à la manière dont les cinq premiers indices sont calculés dans le programme de Gordon.

Généralement, les règles de décision ne sont pas appliquées à tous les niveaux de la hiérarchie mais uniquement aux derniers. Dans nos applications, nous nous sommes resteints aux vingt derniers indices.

4.3 Représentation des résultats

Nous synthétiserons les résultats fournis par les deux programmes à l'aide de tableaux. Ces derniers sont identiques à ceux utilisés par P. André et J.-F. Deschamps. Ils ont la forme suivante :

Nom du jeu de données		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum									
Saut maximum									
Moyenne									
Ward									

TAB. 4.2 – Tableau utilisé pour représenter les résultats

Chaque ligne correspond à une méthode de classification et les huit dernières colonnes se rapportent aux huit méthodes de détermination du nombre de classes qui ont été rappelées dans l'introduction de ce chapitre. Nous noterons par exemple un "2" à l'intersection de la ligne saut minimum et de la colonne m1, si la méthode de Calinski et Harabasz associée à la classification obtenue par la méthode du saut minimum détermine qu'il y a deux classes dans les données.

La deuxième colonne, quant à elle, indique si les classes naturelles sont retrouvées par la méthode de classification lorsqu'on fixe le nombre de classes au nombre de classes réellement présentes dans les données. Un "+" signifie que la méthode les retrouve et un "-" la situation contraire.

4.4 Présentation des principaux résultats obtenus en dimensions 2 et 3 ([1]) ([7])

Dans cette partie, nous allons présenter quelques constatations générales que P. André ([1]) et J.-F. Deschamps ([7]) ont formulées à la suite de la comparaison des six meilleures méthodes de Milligan et Cooper avec celles basées sur le critère des hypervolumes en dimension 2. Nous exposerons également celles émises par J.-F. Deschamps à la suite de la comparaison des six méthodes de Milligan et Cooper sur des ensembles de données en dimension 3. Notons également que pour les ensembles dont nous avons généralisé la structure, nous accompagnerons les résultats en dimension 4 d'un bref commentaire sur les performances des

méthodes en dimensions 2 et 3. Des renseignements plus complets ou relatifs à d'autres ensembles de données en dimensions 2 et 3 sont disponibles dans les mémoires de P. André ([1]) et de J.-F. Deschamps ([7]).

4.4.1 Principales constatations

Tout d'abord, P. André et J.-F. Deschamps ont souligné toute l'importance de fournir une bonne classification aux méthodes de détermination du nombre de classes. En effet, de leurs applications, il ressort que peu de méthodes donnent des résultats corrects quand elles sont appliquées à une méthode hiérarchique ne retrouvant pas les classes naturelles, et même si dans ce cas elles déterminent le bon nombre de classes, les groupes trouvés n'ont pas beaucoup de sens. Par conséquent, il apparaît indispensable de tenir compte des biais des méthodes de classification dans le problème de détermination du nombre de classes.

Pour les méthodes de détermination du nombre de classes basées sur le critère des hypervolumes, J.-F. Deschamps a constaté que lorsque les méthodes hiérarchiques auxquelles elles étaient appliquées, retrouvaient les classes naturelles, elles déterminaient toujours le bon nombre de classes. Il a également observé que dans la situation contraire, elles concluaient chaque fois à l'absence de structure. Cette constatation a été faite en dimension 2 pour des ensembles de données dont certains présentaient une structure particulière.

En ce qui concerne les méthodes de Milligan et Cooper, P. André et J.-F. Deschamps ont remarqué que lorsque les données étaient bien séparées et que chaque classe comptait environ le même nombre d'éléments, ces méthodes fonctionnaient bien. Par contre, une fois qu'elles étaient appliquées à des ensembles de données présentant une structure plus particulière, ils ont constaté qu'elles éprouvaient beaucoup de difficultés à déterminer le nombre correct de classes. Il a été également observé que certaines de ces méthodes donnaient de meilleurs résultats en dimension 3 qu'en dimension 2 et que dès lors, la classement de Milligan et Cooper se confirmait mieux en dimension 3.

Lors du passage de la dimension 2 à la dimension 3 , J.-F. Deschamps a également observé que le nombre de classes déterminé par les méthodes semblait augmenter.

4.5 Analyse des résultats en dimension 4

4.5.1 Données avec trois classes bien séparées

Cet ensemble de données comporte 225 points répartis dans trois classes qui sont bien séparées.

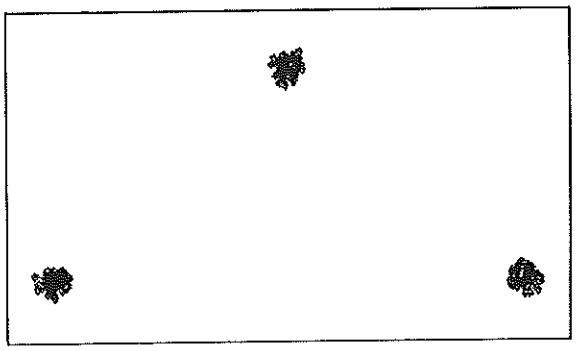


FIG. 4.1 – Projection des données sur le plan principal d'une analyse en composantes principales

On obtient les résultats suivants :

3 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	3	3	3	3	3	3
Saut maximum	+	3	3	3	3	3	3	3	3
Moyenne	+	3	3	3	3	3	3	3	3
Ward	+	3	3	3	3	3	3	3	3

TAB. 4.3 – Résultats obtenus pour l'ensemble de données avec trois classes bien séparées en dimension 4

Nous constatons tout d'abord que les quatre méthodes hiérarchiques retrouvent les classes naturelles, les méthodes de détermination du nombre de classes ont donc été appliquées à de bonnes classifications. Nous voyons également que chacune de ces méthodes retrouve le bon nombre de classes qui est de trois.

Profitons de ce premier ensemble de données pour examiner la façon dont nous avons obtenu les résultats du tableau 4.3. Commençons par les méthodes basées sur le critère des hypervolumes. Pour la méthode du coude (M1), les valeurs obtenues pour le critère des hypervolumes sont les suivantes :

Saut minimum	W(P,k)
k=1	1.08937703 E+10
k=2	1.09544847 E+9
k=3	117611025
k=4	115976507
k=5	115114678
k=6	114361149

TAB. 4.4 – Valeurs du critère des hypervolumes pour la méthode du saut minimum

Rappelons que pour cette méthode, on recherche un coude dans le graphe du critère de classification en fonction du nombre de classes, ce qui revient à rechercher la ligne où la valeur du critère des hypervolumes se stabilise. Cela a lieu en k=3.

Pour la méthode du test du quotient de vraisemblance (M2), on obtient les résultats suivants :

Moyenne	S(k)
k=1	/
k=2	0.10
k=3	0.10
k=4	0.88
k=5	0.90

TAB. 4.5 – Valeurs du quotient de vraisemblance pour la méthode de la moyenne

Pour cette méthode, on recherche la première valeur pour laquelle on rejette l'hypothèse H_0 qui correspond à la présence de k classes ; cette valeur doit être proche de 1. Pour l'ensemble de données traité ici, on constate qu'à partir de k=4, la valeur de S(k) est proche de 1, le nombre de classes présentes dans les données est donc de k-1 qui vaut 3.

Passons à présent, à l'analyse des tableaux contenant les valeurs des indices de Milligan et Cooper. Voici le tableau obtenu pour la méthode de Ward :

Ward	m1	m2	m3	m4	m5	m6
k=8	6349.72	1.8001	0.00054	0.8991	1.2903	57.46
k=7	6803.55	2.3262	0.00052	0.8991	1.3372	59.76
k=6	7511.02	1.0975	0.00052	0.8928	0.7862	62.84
k=5	8435.60	3.1752	0.00042	0.9383	1.4786	67.31
k=4	10127.24	0.6584	0.00031	0.9510	0.6026	74.36
k=3	**13755.78**	**1.8352**	**0.00000**	**1.0000**	**0.9556**	**87.39**
k=2	585.49	15.9312	0.01842	1.0000	118.5375	2.45
k=1	/	13.4229	/	/	6.2430	0.00

TAB. 4.6 – Valeurs des indices de Milligan et Cooper

Rappelons les règles utilisées pour déterminer le nombre de classes. Pour la méthode de Calinski et Harabasz (m1), on recherche la valeur maximale de l'indice, le nombre k de la ligne pour laquelle cette valeur est atteinte est le nombre de classes. On procède également de cette façon pour la méthode CCC (m6). Pour les méthodes de Duda et Hart (m2) et de Beale (m5), basées sur des tests d'hypothèses jugeant de la pertinence ou non de la fusion de deux classes, on recherche la ligne pour laquelle l'indice est supérieur à la valeur de rejet : 3.20 ou 4 pour m2 et 2.37 pour m5. Supposons que cette valeur soit atteinte pour k=t, le nombre de classes est alors de t+1. Pour la méthode du C-index (m3), on recherche la valeur minimale de l'indice, celle-ci étant bornée inférieurement par 0. Finalement, pour la méthode Gamma (m4), on recherche la valeur maximale de l'indice, celle-ci devant être la plus proche possible de sa borne supérieure 1. L'application de toutes ces règles, nous amène donc à choisir comme nombre de classes k=3.

Remarquons que pour la méthode m4, l'indice vaut 1 pour deux valeurs de k : 2 et 3. Ceci est dû au fait que les classes sont fortement éloignées. Tout comme Milligan et Cooper, nous avons choisi le nombre de classes qui donnait le meilleur résultat.

4.5.2 Données avec deux classes bien séparées

Cet ensemble est composé de 160 points répartis dans deux classes. Nous avons considéré cet exemple car dans leur étude, Milligan et Cooper ont constaté que la plupart des méthodes de détermination donnaient leurs plus mauvais résultats quand le nombre de classes était de deux.

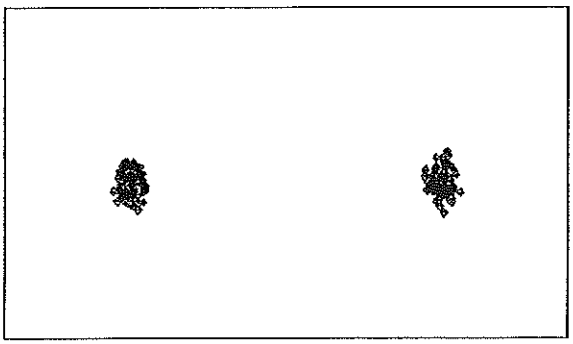


FIG. 4.2 – Projection des données sur le plan principal d'une analyse en composantes principales

Les résultats obtenus sont les suivants :

2 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	2	2

TAB. 4.7 – Résultats obtenus pour l'ensemble de données avec deux classes bien séparées en dimension 4

Les résultats sont très semblables à ceux de l'ensemble de données précédent. Les méthodes basées sur le critère des hypervolumes et celles issues du classement de Milligan et Cooper retrouvent le nombre correct de classes. De plus, ces dernières sont naturelles.

Nous constatons donc que lorsque les classes sont hypersphériques et bien séparées, les méthodes de détermination du nombre de classes n'ont aucun pro-

blème à retrouver le nombre de classes présentes dans les données. Rappelons que cette constatation a déjà été faite pour ce type de données en dimensions 2 et 3.

4.5.3 Données avec deux classes parallèles

Cet ensemble contient 200 points répartis en deux classes parallèles.

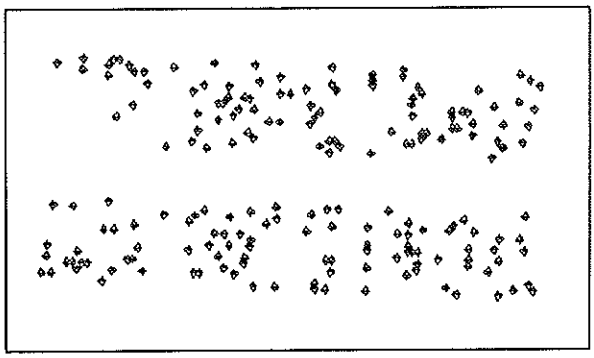


FIG. 4.3 – Projection des données sur le plan principal d'une analyse en composantes principales

Voici les résultats :

2 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	15	8	1	15	1
Saut maximum	-	1	1	2	8 ou 7	2	10	8	8
Moyenne	-	1	1	2	9 ou 7	13	13	7	10
Ward	-	1	1	2	8 ou 7	2	10	8	10

TAB. 4.8 – Résultats obtenus pour l'ensemble de données avec deux classes parallèles en dimension 4

Seule une méthode de classification retrouve les classes naturelles, il s'agit de la méthode du saut minimum. Les trois autres méthodes qui ont pour biais principal de former des classes hypersphériques, ont privilégié la formation de telles classes.

Commençons par analyser la ligne correspondant à la méthode du saut minimum. Nous constatons tout d'abord que les deux méthodes de détermination du

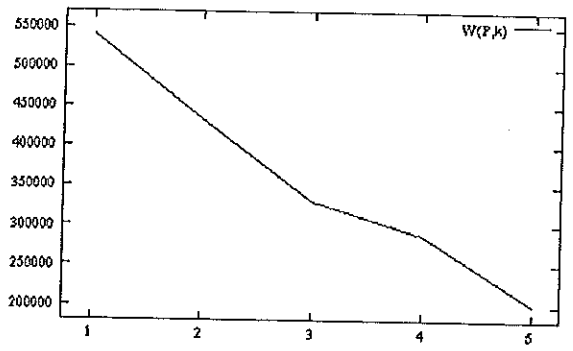


FIG. 4.4 – Méthode du coude - Critère des hypervolumes en fonction du nombre de classes

nombre de classes basées sur le critère des hypervolumes fournissent le nombre correct de classes. Comme en dimension 2, ces deux méthodes semblent donc donner également d'excellents résultats lorsqu'elles sont combinées à une méthode de classification qui retrouve les classes naturelles et qui est différente de la méthode des hypervolumes. Nous remarquons également que comme en dimension 3, la méthode m1 (Calinski et Harabasz) détermine le bon nombre de classes. Les autres méthodes donnent toutes de mauvais résultats.

Passons à présent à l'analyse des lignes correspondant aux méthodes hiérarchiques ne trouvant pas les classes naturelles. Pour les méthodes basées sur le critère des hypervolumes, nous voyons que dans ce cas, comme en dimension 2, toutes deux concluent à l'absence de structure. Ceci résulte du fait que comme les trois méthodes hiérarchiques privilégient des classes hypershériques, l'hypervolume "enlevé" en passant d'une à deux classes n'est pas suffisamment important. Nous pouvons le constater en regardant le tableau 4.9 et le graphique 4.4.

Moyenne	W(P,k)	S(k)
k=1	541067.708	/
k=2	434139.208	0.80
k=3	329603.958	0.75
k=4	288444.5	0.87
k=5	201267.5	0.69

TAB. 4.9 – Valeurs du critère des hypervolumes et du test du quotient de vraisemblance pour la méthode de la moyenne

Examinons maintenant les résultats fournis par les méthodes de Milligan et Cooper. Signalons tout d'abord que pour la méthode de Duda et Hart (m2), nous avons indiqué deux possibilités pour le nombre de classes. Celles-ci dépendent de la valeur critique α choisie pour le test d'hypothèses. Rappelons que Milligan et Cooper ont utilisé un α valant 3.20 alors que Gordon a choisi un α valant 4. Quand nous indiquerons dans le tableau de résultats deux valeurs, la première correspondra toujours au choix de Milligan et Cooper et la seconde à celui de Gordon. En outre, nous constatons que m1 retrouve le bon nombre de classes même quand ces dernières ne sont pas naturelles. C'est le cas également de la méthode du C-index (m3) combinées avec la méthode du saut maximum et de Ward. Pour les méthodes restantes, le nombre de classes déterminé est incorrect. Nous constatons que ces nombres sont relativement élevés.

Comme en dimension 2, cet ensemble de données permet d'illustrer un des principaux biais de la méthode m4. Rappelons qu'à chaque étape de la hiérarchie, l'indice suivant est calculé :

$$\gamma = 1 - \alpha_l$$

où α_l représente une proportion de paires d'objets appartenant à des groupes différents qui sont plus proches que des paires d'objets appartenant au même groupe.

Etant donné la structure des données quand les classes naturelles sont retrouvées, α_l est élevé et donc γ est éloigné de la valeur recherchée 1. Nous pouvons donc affirmer, comme cela a déjà été fait en dimension 2, que la méthode Gamma éprouve des difficultés quand les classes naturelles sont allongées.

Cet ensemble avait également permis à P. André et à J.-F. Deschamps de mettre en évidence un biais de la méthode m1. En effet, en dimension 2, cette méthode déterminait qu'il y avait une classe quand les classes naturelles étaient retrouvées et deux classes dans la situation contraire. A la suite de cette constatation, P. André et J.-F. Deschamps avaient déduit que la méthode de Calinski et Harabasz privilégiait les classes hypersphériques. L'ensemble de données utilisé ici semble contredire leur conclusion. En effet, la méthode détermine le nombre correct de classes quand celles-ci sont naturelles, c'est-à-dire quand les deux classes allongées sont retrouvées.

4.5.4 Données avec trois classes parallèles

Cet ensemble contient 300 points répartis en trois classes parallèles.

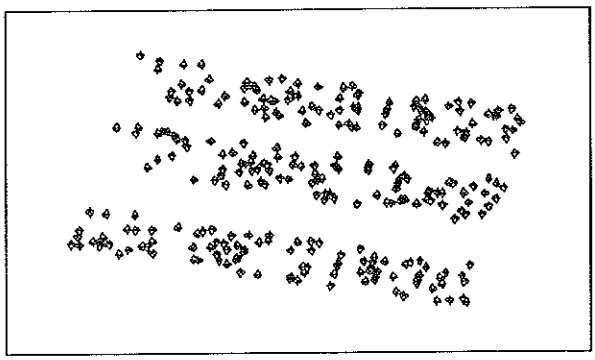


FIG. 4.5 – Projection des données sur le plan principal d'une analyse en composantes principales

On obtient les résultats suivants :

Trois classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	2	17	10	2	17	1
Saut maximum	-	1	1	4	14 ou 9	16	2	14	13
Moyenne	-	1	1	4	12 ou 11	18	2	12	9
Ward	-	1	1	4	12 ou 10	16	2	12	15

TAB. 4.10 – Résultats obtenus pour l'ensemble de données avec trois classes parallèles en dimension 4

Tout d'abord, nous constatons que comme pour l'ensemble précédent, seule la méthode du saut minimum retrouve les classes naturelles.

Les conclusions pour les méthodes basées sur le critère des hypervolumes sont identiques : quand les classes naturelles sont retrouvées, les deux méthodes déterminent le nombre correct de classes; dans le cas contraire, elles concluent à l'absence de structure. Suite à ces conclusions et celles obtenues pour l'ensemble de données précédent, nous pouvons réaffirmer ici toute l'importance de fournir aux méthodes de détermination du nombre de classes une classification contenant les classes naturelles.

Contrairement aux résultats obtenus pour l'ensemble de données précédent, aucune des méthodes de Milligan et Cooper ne fournit des résultats corrects.

Remarque

A la suite de la remarque concernant la méthode m4 faite pour l'ensemble de données avec deux classes parallèles, on s'attendrait à observer un résultat semblable pour les données traitées ici. Or, on constate que combinée avec la méthode du saut minimum, on ne conclut pas à l'absence de structure mais à la présence de deux classes. Ce résultat s'explique : il est dû à la structure de l'ensemble de données. En effet, lors du passage de trois à deux classes, c'est-à-dire lors de la fusion de la classe centrale avec une autre classe, la proportion d'objets "mal classés" diminue. La valeur de l'indice est donc plus grande en k=2.

4.5.5 Données "gros-petit" éloignés

Cet ensemble de données contient 190 points, sa principale caractéristique est qu'il est composé de deux classes d'effectifs très différents. La première est composée de 150 points et la seconde de 40.

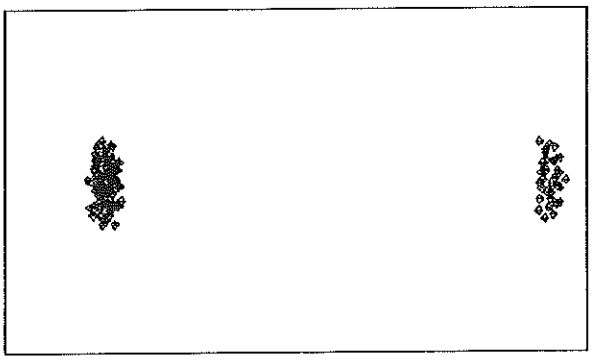


FIG. 4.6 – Projection des données sur le plan principal d'une analyse en composantes principales

On obtient les résultats suivants :

gros-petit éloignés		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	9	2
Saut maximum	+	2	2	2	2	2	2	18	2
Moyenne	+	2	2	2	2	2	2	10	2
Ward	+	2	2	2	2	2	2	14	2

TAB. 4.11 – Résultats obtenus pour l'ensemble de données "gros-petit" éloignés en dimension 4

Toutes les méthodes de classification retrouvent les classes naturelles, les méthodes de détermination du nombre de classes ont par conséquent été utilisées dans de bonnes conditions.

A la vue du tableau 4.11, nous constatons que toutes les méthodes à l'exception de celle de Beale déterminent le bon nombre de classes. Remarquons que ces résultats sont normaux puisque la structure de cet ensemble diffère peu de celle de l'ensemble de données avec deux classes séparées, le seul changement étant la

différence d'effectifs. Celui-ci semble donc être fatal pour la méthode de Beale. Rappelons qu'en dimension 2, cette différence faisait échouer deux méthodes : celle de Beale et celle du CCC. Des résultats identiques ont été observés en dimension 3.

4.5.6 Données "gros-petit" proches

Cet ensemble comporte 190 points répartis en deux classes contenant respectivement 150 et 40 points, qui sont proches l'une de l'autre.

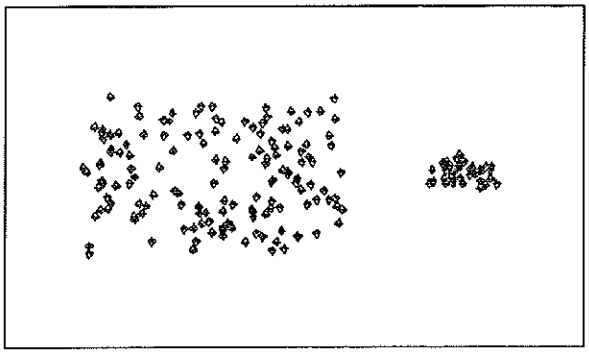


FIG. 4.7 – Projection des données sur le plan principal d'une analyse en composantes principales

Les résultats obtenus sont les suivants :

Gros-petit proches		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	1	1	1
Saut maximum	+	2	2	2	2	2	1	1	7
Moyenne	+	2	2	2	2	2	1	1	9
Ward	+	2	2	2	2	2	1	1	6

TAB. 4.12 – Résultats obtenus pour l'ensemble de données "gros-petit" proches en dimension 4

Nous constatons tout d'abord que les quatre méthodes hiérarchiques que nous avons utilisées retrouvent les classes naturelles.

Examinons à présent les résultats obtenus par les méthodes de détermination du nombre de classes. Nous remarquons que les méthodes basées sur le critère des hypervolumes ainsi que les méthodes de Calinski et Harabasz (m1), de Duda et Hart (m2) et du C-index (m3) déterminent le nombre correct de classes. Rappelons que ces trois dernières méthodes ont été classées respectivement première, deuxième et troisième par Milligan et Cooper. Les trois autres méthodes donnent, quant à elles, des résultats incorrects. Il semble donc que rapprocher les classes soit source de difficultés pour ces méthodes. Rappelons qu'en dimension 2, les méthodes m2, m5 et m6 déterminaient un nombre incorrect de classes. Ce résultat a également été observé en dimension 3.

Remarquons que pour cet ensemble de données, la méthode Gamma conclut à l'absence de structure. Les raisons de ce résultat sont les mêmes que celles évoquées pour l'ensemble de données avec deux classes parallèles : beaucoup de paires de points appartenant à des classes différentes sont plus proches que des paires de points appartenant à la même classe. La méthode m4 semble donc avoir également des difficultés quand les classes sont proches l'une de l'autre.

Suite aux résultats obtenus pour cet ensemble de données en dimension 3, J.-F. Deschamps en avait déduit que la méthode de Duda et Hart ne tenait pas compte des classes de petit effectif. Ceci est contredit ici puisque la méthode détermine le nombre correct de classes.

4.5.7 Données sans structure

Cet ensemble est composé de 110 points.

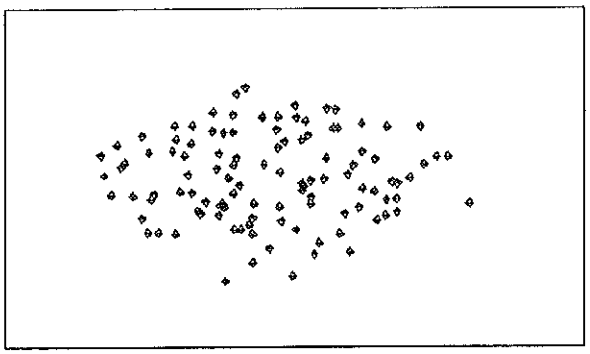


FIG. 4.8 – Projection des données sur le plan principal d'une analyse en composantes principales

Nous obtenons les résultats suivants :

Sans structure	M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	1	1	2	1	1	1	1	1
Saut maximum	1	1	4	1	1	1	1	1
Moyenne	1	1	4	1	1	1	1	1
Ward	1	1	4	1	1	1	1	1

TAB. 4.13 – Résultats obtenus pour l'ensemble de données sans structure en dimension 4

Toutes les méthodes détectent l'absence de structure dans les données, à l'exception de celle de Calinski et Harabasz qui pourtant est classée première par Milligan et Cooper. Un résultat similaire a déjà été observé en dimension 2.

4.5.8 Données avec un pont entre les classes

Cet ensemble comporte 189 points. Il est composé de deux classes contenant chacune 80 points. Ces deux classes sont reliées entre elles par un pont composé de 29 points.

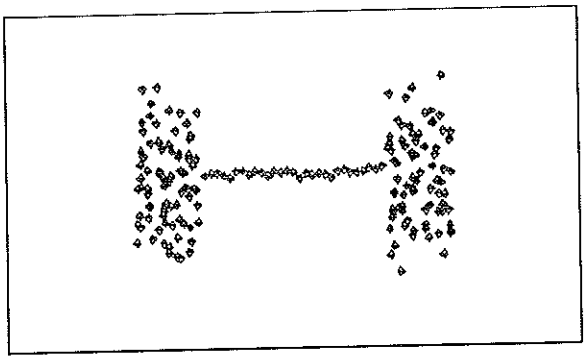


FIG. 4.9 – Projection des données sur le plan principal d'une analyse en composantes principales

Nous obtenons les résultats suivants :

Pont		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	-	1	1	2	1	1	1	1	1
Saut maximum	+	2	2 ou 3	3	12 ou 4	3	3	12	2
Moyenne	+	2	2 ou 3	3	11 ou 3	3	3	11	2
Ward	+	2	2 ou 3	3	12 ou 4	3	3	12	2

TAB. 4.14 – Résultats obtenus pour l'ensemble de données avec un pont entre les classes en dimension 4

Signalons tout d'abord que nous avons indiqué un "+" dans la deuxième colonne quand la méthode de classification retrouvait les deux "grosses" classes lorsqu'on fixe le nombre de classes à deux. Remarquons que contrairement aux ensembles de données traités précédemment, celui-ci n'a pas une structure qui permet de distinguer clairement la frontière des classes. C'est pourquoi, nous préférons ne pas parler ici de classes naturelles. De plus, nous ne savons pas exactement si le nombre de classes dans les données est de deux ou de trois.

Seule la méthode du saut minimum ne retrouve pas les deux classes. L'application de cette méthode à un ensemble de données comme celui-ci permet d'illustrer l'effet de chaînage. En effet, cette méthode préfère, à chaque étape de la hiérarchie, ajouter un point à la classe existante au lieu d'en créer une nouvelle. C'est pourquoi, la partition en deux classes se compose d'un ensemble contenant 188 points et d'un autre contenant un seul point. Cette partition permet d'expliquer pourquoi toutes les méthodes sauf celle de Calinski et Harabasz (m1) concluent à l'absence de structure.

Examinons les résultats obtenus en utilisant les trois autres méthodes de classification. Les méthodes de détermination du nombre de classes basées sur le critère des hypervolumes déterminent deux ou trois comme nombre de classes. Voici la valeur des indices pour ces méthodes et le graphe associé pour la méthode du coude :

Saut maximum	W(P,k)	S(k)
k=1	320136557	/
k=2	118460702	0.37
k=3	89480609.3	0.75
k=4	89476182.7	0.99
k=5	75389223.6	0.84

TAB. 4.15 – Valeurs du critère des hypervolumes et du test du quotient de vraisemblance pour la méthode du saut maximum

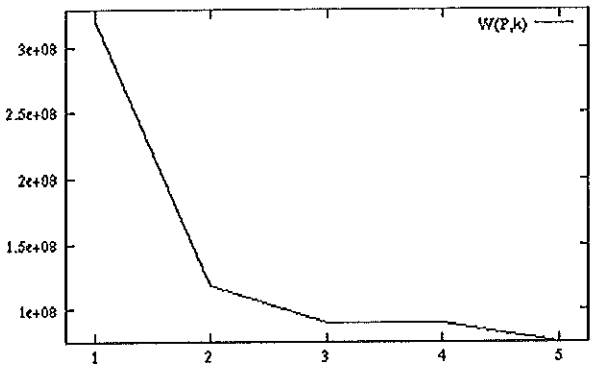


FIG. 4.10 – Methode du coude - Critère des hypervolumes en fonction du nombre de classes

Le choix entre deux et trois classes pour le test du quotient de vraisemblance est difficile, nous obtenons déjà une valeur assez proche de 1 pour le quotient de vraisemblance en k=3 et encore une plus proche en k=4.

Pour les méthodes de Milligan et Cooper, m1, m3, m4 et m6 donnent des résultats satisfaisants. Pour m2, avec $\alpha = 4$ et combinée avec la méthode de la moyenne, on obtient également un résultat satisfaisant. La méthode de Beale quant à elle, semble avoir de grosses difficultés.

Remarquons que cet ensemble de données est le seul pour lequel le choix de la valeur de α pour m2 influence le résultat. Un seul ensemble de données est évidemment trop peu pour favoriser l'une ou l'autre valeur de α.

4.5.9 Les iris de Fisher

Contrairement aux ensembles de données traités jusqu'à présent, ces données sont réelles. Elles se composent de 150 fleurs provenant de trois variétés d'iris : setosa, versicolor, virginica. Sur chacune des plantes, quatre caractéristiques ont été observées, il s'agit de la longueur et de la largeur des pétales et des sépales. Chaque variété est représentée par 50 fleurs.

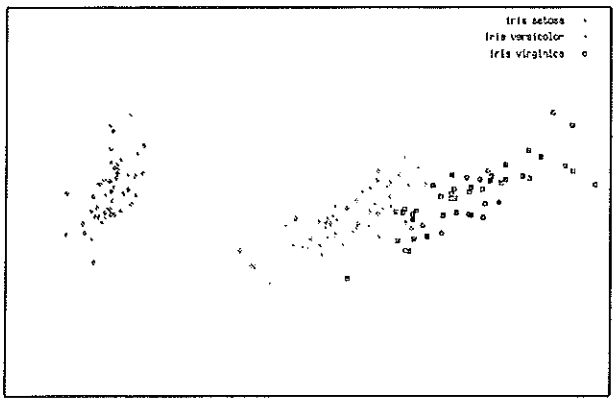


FIG. 4.11 – Projection des données sur le plan principal d'une analyse en composantes principales

En appliquant à ces données les différentes méthodes, on obtient les résultats suivants :

Iris de Fisher		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	3	2	2	2
Saut maximum	*	4	4	3	5 ou 4	5	2	18	6
Moyenne	+	4	4	3	7 ou 3	15	2	15	2
Ward	+	4	4	3	6 ou 4	5	2	5	3

TAB. 4.16 – Résultats obtenus pour les iris de Fisher

Pour commencer, signalons que nous avons mis un "+" dans la deuxième colonne quand la partition en deux classes se composait d'une part des iris setosa et d'autre part des deux autres espèces. Les méthodes du saut minimum, de la moyenne et de Ward trouvent une telle partition. Pour la méthode du saut maximum, les partitions en deux classes obtenues par SAS et par le programme de Gordon sont différentes, celle du programme de Gordon sépare les iris setosa des deux autres variétés. Nous avons symbolisé cette situation en mettant le caractère "*" dans la deuxième colonne. Rappelons que la classification provenant de SAS est utilisée par les méthodes M1, M2 et m6.

Remarquons qu'aucune méthode de classification n'est capable de fournir une partition en trois classes où chaque variété d'iris est une classe. Cette partition se compose plutôt du groupe des iris setosa et de deux autres groupes où les iris versicolor et virginica sont mélangés. Signalons que si l'on considère que chaque variété d'iris forme une classe, les enveloppes convexes des classes correspondant aux iris versicolor et virginica ne sont pas disjointes. Une des hypothèses des méthodes de détermination du nombre de classes basées sur le critère des hypervolumes n'est donc pas vérifiée.

Passons à présent à l'analyse des résultats fournis par les méthodes de détermination du nombre de classes. Commençons par ceux obtenus en utilisant la méthode du saut minimum. Nous constatons que la plupart des méthodes déterminent deux comme nombre de classes. Ce résultat rejoint l'avis de plusieurs auteurs qui affirment qu'il y a deux classes dans les données.

Pour les méthodes basées sur le critère des hypervolumes et associées aux méthodes de la moyenne et de Ward, nous constatons que malgré la partition

en deux classes, ces méthodes déterminent quatre comme nombre de classes présentes dans les données. Cette différence avec les résultats obtenus par la méthode du saut minimum est liée aux biais des méthodes de classification. En effet, la partition en trois classes obtenues par la méthode du saut minimum contient un groupe formé des iris setosa, un groupe où les iris versicolor et virginica sont mélangés et un groupe constitué de deux points. Comme on peut le constater à la vue du tableau 4.17 l'hypervolume enlevé en passant de deux à trois classes n'est pas très grand ; c'est pourquoi l'application des règles de décision nous amène à conclure qu'il y a deux classes dans les données. Par contre, pour les méthodes de la moyenne et de Ward, le passage de deux à trois classes a pour effet de diviser la classe où les espèces versicolor et virginica sont mélangées de telle sorte qu'il y ait un gain d'hypervolume plus important, ceci peut être constaté en observant les valeurs de W(P,k) dans le tableau 4.18. Le passage de trois à quatre classes a le même effet et il faut attendre la partition en 5 classes pour la méthode de la moyenne pour observer une stabilisation de la valeur du critère des hypervolumes. Notons que la partition en quatre classes obtenue en utilisant la méthode de la moyenne se compose du groupe des iris setosa, d'un groupe où les iris versicolor et virginica sont mélangés, d'un groupe d'iris virginica et d'un groupe d'iris versicolor.

Saut minimum	W(P,k)	S(k)
k=1	4.6810375	/
k=2	1.88514997	0.40
k=3	1.62917087	0.86
k=4	1.52691247	0.93
k=5	1.37771247	0.90

TAB. 4.17 – Valeurs du critère des hypervolumes et du test du quotient de vraisemblance pour la méthode du saut minimum

Moyenne	W(P,k)	S(k)
k=1	4.6810375	/
k=2	1.88514997	0.40
k=3	1.19809587	0.63
k=4	0.7737125	0.64
k=5	0.755662497	0.97

TAB. 4.18 – Valeurs du critère des hypervolumes et du test du quotient de vraisemblance pour la méthode de la moyenne

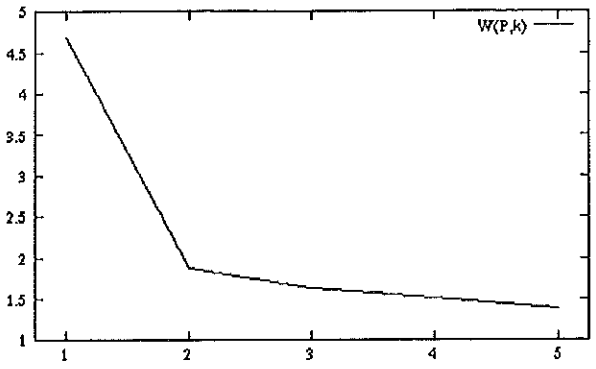


FIG. 4.12 – Méthode du coude - Critère des hypervolumes en fonction du nombre de classes

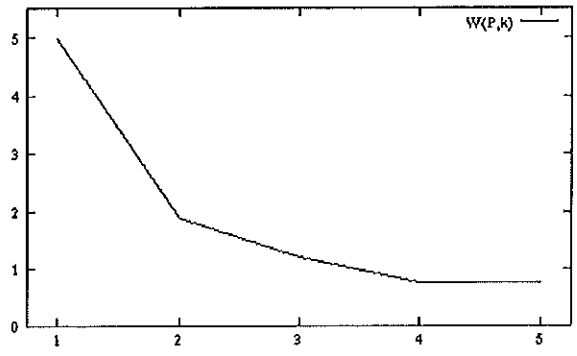


FIG. 4.13 – Methode du coude - Critère des hypervolumes en fonction du nombre de classes

L'application de ces deux méthodes à la classification obtenue par la méthode du saut maximum nous amène également à déterminer qu'il y a quatre classes dans les données. La partition correspondante est du même genre que celle obtenue par la méthode de la moyenne.

Pour ce qui est des méthodes de Milligan et Cooper associées à ces trois méthodes de classification, nous constatons que la méthode m4 conclut à la présence de deux classes. Ceci est également observé pour la méthode du CCC combinée avec celle de la moyenne. Les autres méthodes de détermination du nombre de classes donnent d'autres résultats.

4.6 Analyse des résultats en dimension 5

4.6.1 Données avec trois classes bien séparées

Cet ensemble de données comprend 240 points répartis en trois classes. Voici les résultats :

3 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	3	3	3	3	3	3
Saut maximum	+	3	3	3	3	3	3	3	3
Moyenne	+	3	3	3	3	3	3	3	3
Ward	+	3	3	3	3	3	3	3	3

TAB. 4.19 – Résultats obtenus pour l'ensemble de données avec trois classes bien séparées en dimension 5

Nous constatons que les quatre méthodes de classification retrouvent les classes naturelles, les méthodes de détermination du nombre de classes ont donc été combinées à de bonnes classifications. De plus, toutes ces méthodes déterminent le nombre correct de classes qui est de trois.

4.6.2 Données avec deux classes bien séparées

Il s'agit d'un ensemble de données contenant 180 points répartis en deux classes qui sont également bien séparées. Nous obtenons les résultats suivants :

2 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	2	2

TAB. 4.20 – Résultats obtenus pour l'ensemble de données avec deux classes bien séparées en dimension 5

Comme pour l'ensemble de données précédent, les méthodes de détermination retrouvent le nombre correct de classes qui sont naturelles.

Profitons de cet ensemble de données pour examiner les valeurs prises par les indices de Milligan et Cooper quand les classes sont bien séparées. Le programme de Gordon nous fournit pour la méthode du saut maximum, le tableau suivant :

saut maximum	m1	m2	m3	m4	m5	m6
k=6	3652.76	1.9952	0.0015	0.7787	1.3939	13.720
k=5	4151.03	1.7657	0.0016	0.7535	1.0886	16.032
k=4	5140.54	0.7833	0.0015	0.7712	0.6670	19.578
k=3	7333.40	-0.2121	0.0011	0.8290	0.3745	26.323
k=2	**13360.75**	**0.6490**	**0.0000**	**1.0000**	**0.5990**	**41.932**
k=1	/	18.8120	/	/	257.920	0.000

TAB. 4.21 – Valeurs des indices des six méthodes de Milligan et Cooper appliquées à la classification obtenue en utilisant la méthode du saut maximum

En appliquant à chacune de ces six méthodes la règle de décision correspondante, nous constatons que toutes arrivent à la même conclusion : il y a deux classes dans les données.

Il est à remarquer que lorsque les données ont une structure bien nette, les indices ont des valeurs "claires" qui permettent facilement d'opter pour un nombre de classes. Par exemple, pour la méthode Gamma (m4), nous constatons que l'indice atteint, pour k=2, sa valeur maximale qui est de 1. Pour la méthode du C-index (m3), dont la règle de décision consiste à rechercher la valeur minimale de l'indice, nous remarquons que celle-ci est atteinte également pour k=2 avec la plus petite valeur possible : 0. Nous verrons, plus tard, que pour des ensembles de données présentant une structure plus particulière, comme les données avec des classes parallèles, de telles valeurs ne sont plus observées. Atteindre ces valeurs pour un ensemble de données dont la structure est inconnue peut donc permettre à l'utilisateur de déterminer sans crainte le nombre de classes.

4.6.3 Données avec deux classes parallèles

Cet ensemble de données comprend 220 points répartis en deux classes qui sont parallèles. Voici les résultats obtenus :

2 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	11	1	1	11	1
Saut maximum	-	1	1	2	9 ou 8	13	10	9	7
Moyenne	-	1	1	2	8	15	11	8	9
Ward	-	1	1	2	8	12	10	8	9

TAB. 4.22 – Résultats obtenus pour l'ensemble de données avec deux classes parallèles en dimension 5

Comme dans les dimensions inférieures, seule la méthode du saut minimum retrouve les classes naturelles. Remarquons que les deux méthodes de détermination du nombre de classes basées sur le critère des hypervolumes associées à cette classification trouvent le nombre correct de classes. Ceci est également observé pour la méthode de Calinski et Harabasz (m1) classée première par Milligan et Cooper. Nous constatons aussi que comme en dimension 4, la méthode Gamma (m4) associée à la méthode du saut minimum conclut à l'absence de structure.

Quand les classes naturelles ne sont pas retrouvées, l'hypervolume enlevé en passant d'une à deux classes est minime, c'est pourquoi les méthodes M1 et M2 concluent à l'absence de structure. La méthode m1, quant à elle, retrouve également dans ce cas, le nombre correct de classes.

Notons que les méthodes m2, m3, m4, m5 et m6 associées à n'importe quelle méthode hiérarchique déterminent un nombre incorrect de classes.

A présent, examinons la valeur des indices de Milligan et Cooper obtenus en appliquant les méthodes de détermination à la hiérarchie de partitions fournie par la méthode du saut minimum. Ceux-ci sont répertoriés dans le tableau 4.23. Nous constatons que pour m3 et m4, les valeurs des indices permettant de déterminer le nombre de classes sont beaucoup moins évidentes que pour l'ensemble de données avec deux classes bien séparées. En effet, des valeurs telles que 0 pour m3 et 1 pour m4 ne sont plus atteintes dans le cas présent.

Saut minimum	m1	m2	m3	m4	m5	m6
k=12	26.03	-0.6088	0.1402	0.7447	0.1297	-22.89
k=11	27.75	-1.3892	0.1442	0.7459	0.1295	-22.44
k=10	13.84	6.1946	0.2630	0.7631	2.6973	-28.12
k=9	9.98	3.3966	0.3078	0.7676	1.4110	-29.72
k=8	11.10	-1.8979	0.3099	0.7678	0.0633	-28.85
k=7	12.12	-1.5265	0.3176	0.7681	0.1316	-28.00
k=6	14.20	-1.9907	0.3191	0.7682	0.0503	-26.74
k=5	17.15	1.9934	0.3225	0.7684	0.0514	-25.08
k=4	22.87	-2.2677	0.3212	0.7685	0.0064	-23.15
k=3	32.92	-1.8585	0.3248	0.7686	0.0749	-20.52
k=2	62.87	-1.8405	0.3291	0.7686	0.0796	-11.37
k=1	/	2.4731	/	/	0.8859	0.00

TAB. 4.23 – Valeurs des indices des six méthodes de Milligan et Cooper appliquées à la classification obtenue en utilisant la méthode du saut minimum

En ce qui concerne les méthodes basées sur le critère des hypervolumes, le choix du nombre de classes ne pose aucun problème. Par exemple, pour les méthodes M1 et M2 combinées à la méthode du saut maximum, nous obtenons les valeurs suivantes :

Saut maximum	W(P,k)	S(k)
k=1	6631025.84	/
k=2	4753311.11	0.71
k=3	3695255.77	0.77
k=4	2483479.69	0.67
k=5	1360066.53	0.54

TAB. 4.24 – Valeurs du critère des hypervolumes et du test du quotient de vraisemblance pour la méthode du saut maximum

Pour la méthode du coude, nous constatons sur la figure 4.14 que la valeur du critère des hypervolumes ne se stabilise pas. En ce qui concerne M2, on atteint en k=2 une valeur proche de 1 qui n'augmente guère par la suite. Ceci permet par conséquent de conclure à l'absence de structure.

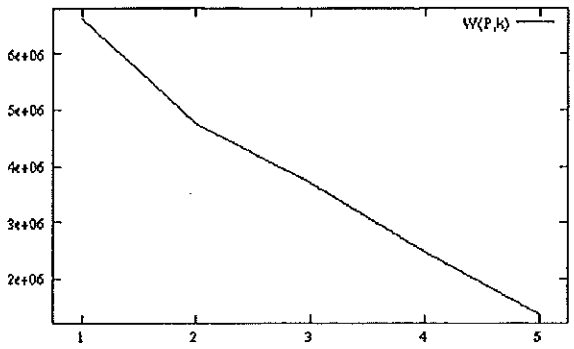


FIG. 4.14 – Méthode du coude - Critère des hypervolumes en fonction du nombre de classes

4.6.4 Données avec trois classes parallèles

Cet ensemble de données comprend 330 points répartis en trois classes parallèles. Nous obtenons les résultats suivants :

3 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	2	18	5	2	18	1
Saut maximum	-	1	1	4	13 ou 11	15	2	13	7
Moyenne	-	1	1	4	12	18	2	12	9
Ward	-	1	1	4	12 ou 11	15	2	12	15

TAB. 4.25 – Résultats obtenus pour l'ensemble de données avec trois classes parallèles en dimension 5

Nous constatons ici que seules les méthodes basées sur le critère des hypervolumes et combinées à la méthode du saut minimum retrouvent le nombre correct de classes. Remarquons également qu'ici la méthode de Calinski et Harabasz appliquée à une méthode de classification fournissant les classes naturelles, ne donne pas le nombre correct de classes alors que cette méthode fonctionnait bien pour l'ensemble de données précédent.

Nous constatons également, comme précédemment, que quand les classes naturelles ne sont pas retrouvées, les méthodes basées sur le critère des hypervolumes concluent à l'absence de structure.

4.6.5 Données "gros-petit" éloignés

Cet ensemble de données comporte 220 points répartis en deux classes d'effectifs différents. La première contient 170 points et la seconde 50. Voici les résultats :

gros-petit éloignés		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	20	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	20	2

TAB. 4.26 – Résultats obtenus pour l'ensemble de données avec "gros-prtit éloignés" en dimension 5

Nous constatons que toutes les méthodes, sauf m5, déterminent le nombre exact de classes. De plus, ces dernières sont naturelles. Cet ensemble de données n'ayant pas une structure tellement différente de l'ensemble de données avec deux classes séparées, il est normal qu'on observe un tel résultat.

4.6.6 Données "gros-petit" proches

Cet ensemble de données se compose, tout comme le précédent, de 220 points répartis en deux classes de respectivement 170 et 50 points. Ces deux dernières présentent la caractéristique d'être relativement proches l'une de l'autre. Nous obtenons les résultats suivants :

gros-petit proches		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	3
Moyenne	+	2	2	2	2	2	2	2	3
Ward	+	2	2	2	2	2	2	2	5

TAB. 4.27 – Résultats obtenus pour l'ensemble de données "gros-petit" proches en dimension 5

Nous remarquons que les méthodes de détermination du nombre de classes basées sur le critère des hypervolumes et celles de Milligan et Cooper, à l'exception de m6, retrouvent le nombre correct de classes qui sont naturelles.

4.6.7 Données sans structure

Cet ensemble contient 125 points. Voici les résultats :

Sans structure	M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	1	1	2	1	1	1	1	1
Saut maximum	1	1	2	1	1	1	1	1
Moyenne	1	1	4	1	1	1	11	1
Ward	1	1	2	1	1	1	1	1

TAB. 4.28 – Résultats obtenus pour l'ensemble de données sans structure en dimension 5

Les résultats obtenus pour cet ensemble de données sont assez semblables à ceux observés pour les dimensions inférieures. Toutes les méthodes à l'exception de celle de Calinski et Harabasz (m1) et celle de Beale (m5) combinée à la méthode de la moyenne concluent à l'absence de structure.

4.6.8 Données avec un pont entre les classes

Cet ensemble comporte 209 points. Il est composé de deux classes contenant chacune 90 points. Ces dernières sont reliées entre elles par un pont de 29 points. Nous obtenons les résultats suivants :

Pont		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	-	1	1	4	1	1	1	1	1
Saut maximum	+	2	2 ou 3	2	19 ou 4	2	2	19	2
Moyenne	+	2	2 ou 3	2	3 ou 2	2	2	10	2
Ward	+	2	2 ou 3	2	19 ou 2	2	2	19	2

TAB. 4.29 – Résultats obtenus pour l'ensemble de données avec un pont entre les classes en dimension 5

Signalons tout d'abord que nous avons rempli la deuxième colonne bien que pour cet ensemble il n'existe pas de classes naturelles évidentes. Nous avons mis le symbole "+" quand les deux classes ont été retrouvées par la méthode de classification quand on fixe le nombre de classes à deux.

Nous constatons que seule la méthode du saut minimum ne retrouve pas les deux classes. Cet ensemble de données permet donc de mettre en évidence son principal biais, à savoir l'effet de chaînage. Suite à la classification obtenue par cette méthode, la plupart des méthodes de détermination concluent à l'absence de structure à l'exception de celle de Calinski et Harabasz. Ce qui est normal vu les résultats obtenus pour l'ensemble de données précédent.

Pour les autres méthodes de classification, nous constatons que la plupart des méthodes de détermination du nombre de classes, à l'exception de celles de Beale et de Duda et Hart combinée aux méthodes du saut maximum et de Ward quand α vaut 3.2, concluent qu'il y a deux ou trois classes dans les données.

Chapitre 5

Comparaison des six meilleures méthodes de détermination du nombre de classes de Milligan et Cooper avec celles basées sur le critère des hypervolumes à partir d'ensembles de données de dimension 6 à 8.

5.1 Introduction

Dans ce chapitre, nous allons poursuivre la comparaison des six premières méthodes de détermination du nombre de classes issues du classement de Milligan et Cooper avec celles basées sur le critère des hypervolumes. Comme précédemment, ces méthodes ont été appliquées aux classifications fournies par les méthodes du saut minimum, du saut maximum, de la moyenne et de Ward pour divers ensembles de données dont la dimension varie cette fois-ci entre 6 et 8.

Les résultats de cette comparaison étant similaires à ceux observés pour les ensembles de données en dimension 4 et 5, nous ne les commenterons pas individuellement, nous en ferons plutôt une synthèse à la fin de ce chapitre.

5.2 Résultats en dimension 6

5.2.1 Données avec trois classes bien séparées

Cet ensemble de données comporte 255 points répartis en trois classes bien séparées. Voici les résultats :

3 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	3	3	3	3	3	3
Saut maximum	+	3	3	3	3	3	3	3	3
Moyenne	+	3	3	3	3	3	3	3	3
Ward	+	3	3	3	3	3	3	3	3

TAB. 5.1 – Résultats obtenus pour l'ensemble de données avec trois classes bien séparées en dimension 6

5.2.2 Données avec deux classes bien séparées

Cet ensemble de données se compose de 180 points répartis en deux classes. Nous obtenons les résultats suivants :

2 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	2	2

TAB. 5.2 – Résultats obtenus pour l'ensemble de données avec deux classes bien séparées en dimension 6

5.2.3 Données avec deux classes parallèles

Cet ensemble de données contient 230 points répartis en deux classes qui présentent la caractéristique d'être parallèles. Nous obtenons les résultats suivants :

2 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	1	1	1	1
Saut maximum	-	1	1	2	6	6	6	6	6
Moyenne	-	1	1	3	6	6	6	19	6
Ward	-	1	1	2	7 ou 5	8	8	5	5

TAB. 5.3 – Résultats obtenus pour l'ensemble de données avec deux classes parallèles en dimension 6

5.2.4 Données avec trois classes parallèles

Cet ensemble de données se compose de 345 points répartis en trois classes parallèles. Nous obtenons les résultats suivants :

3 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	2	17 ou 16	7	2	16	1
Saut maximum	-	1	1	4	15 ou 12	1	2	15	9
Moyenne	-	1	1	9	15	1	2	15	15
Ward	-	1	1	9	15 ou 12	1	2	15	15

TAB. 5.4 – Résultats obtenus pour l'ensemble de données avec trois classes parallèles en dimension 6

5.2.5 Données "gros-petit" éloignés

Cet ensemble de données comporte 240 points répartis en deux classes d'effectifs différents. La première compte 180 points et la seconde 60 points. De plus, ces deux classes sont éloignées l'une de l'autre. Nous obtenons les résultats suivants :

"gros-petit" éloignés		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	11	2
Ward	+	2	2	2	2	2	2	16	2

TAB. 5.5 – Résultats obtenus pour l'ensemble de données "gros-petit" éloignés en dimension 6

5.2.6 Données sans structure

Cet ensemble de données se compose de 130 points. Nous obtenons les résultats suivants :

Sans structure	M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	1	1	2	1	1	1	1	1
Saut maximum	1	1	3	1	1	1	19	1
Moyenne	1	1	2	1	1	1	18	1
Ward	1	1	3	1	1	1	19	1

TAB. 5.6 – Résultats obtenus pour l'ensemble de données sans structure en dimension 6

5.2.7 Données avec un pont entre les classes

Cet ensemble de données se compose de deux classes contenant chacune 95 points et reliées entre elles par un pont de 29 points. Nous obtenons les résultats suivants :

Pont		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	-	1	1	3	1	1	1	1	1
Saut maximum	+	2	2	2	5	3	2	5	2
Moyenne	+	2	2	2	3	3	2	5	2
Ward	+	2	2	2	6	3	2	6	2

TAB. 5.7 – Résultats obtenus pour l'ensemble de données avec un pont entre les classes en dimension 6

5.3 Résultats en dimension 7

5.3.1 Données avec trois classes bien séparées

Il s'agit d'un ensemble de données contenant 270 points répartis en trois classes bien séparées. Nous obtenons les résultats suivants :

3 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	3	3	3	3	3	3
Saut maximum	+	3	3	3	3	3	3	3	3
Moyenne	+	3	3	3	3	3	3	3	3
Ward	+	3	3	3	3	3	3	3	3

TAB. 5.8 – Résultats obtenus pour l'ensemble de données avec trois classes bien séparées en dimension 7

5.3.2 Données avec deux classes bien séparées

Cet ensemble de données se compose de 190 points répartis en deux classes qui sont bien séparées. Nous obtenons les résultats suivants :

2 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	2	2

TAB. 5.9 – Résultats obtenus pour l'ensemble de données avec deux classes bien séparées en dimension 7

5.3.3 Données avec deux classes parallèles

Cet ensemble de données contient 240 points répartis en deux classes qui sont parallèles. Voici les résultats :

2 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	5	1	2	1
Saut maximum	-	1	1	2	6	11	10	6	4
Moyenne	-	1	1	4	7 ou 4	11	11	6	4
Ward	-	1	1	2	6	15	10	6	6

TAB. 5.10 – Résultats obtenus pour l'ensemble de données avec deux classes parallèles en dimension 7

5.3.4 Données avec trois classes parallèles

Il s'agit d'un ensemble de données contenant 360 points répartis en trois classes parallèles. Nous obtenons les résultats suivants :

3 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	2	3 ou 2	5	2	2	1
Saut maximum	-	1	1	10	15 ou 13	17	2	16	10
Moyenne	-	1	1	11	13 ou 12	1	2	17	13
Ward	-	1	1	4	15 ou 13	17	2	16	15

TAB. 5.11 – Résultats obtenus pour l'ensemble de données avec trois classes parallèles en dimension 7

5.3.5 Données "gros-petit" éloignés

Cet ensemble de données contient 250 points. Il se compose de deux classes éloignées l'une de l'autre. La première compte 190 points et la seconde 60. Nous obtenons les résultats suivants :

"gros-petit" éloignés		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	18	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	18	2

TAB. 5.12 – Résultats obtenus pour l'ensemble de données "gros-petit" eloignés en dimension 7

5.3.6 Données sans structure

Cet ensemble de données contient de 135 points. Nous obtenons les résultats suivants :

Sans structure	M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	1	1	2	1	1	1	1	1
Saut maximum	1	1	2	1	1	1	12	1
Moyenne	1	1	6	1	1	1	1	1
Ward	1	1	2	1	1	1	1	1

TAB. 5.13 – Résultats obtenus pour l'ensemble de données sans structure en dimension 7

5.3.7 Données avec un pont entre les classes

Cet ensemble de données se compose de deux classes contenant chacune 95 points et reliées entre elles par un pont de 29 points. Voici les résultats :

Pont		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	-	1	1	2	1	1	1	1	1
Saut maximum	+	2	2	2	12	2	2	12	2
Moyenne	+	2	2 ou 3	2	3	2	2	2	2
Ward	+	2	2 ou 3	2	11	2	2	11	2

TAB. 5.14 – Résultats obtenus pour l'ensemble de données avec un pont entre les classes en dimension 7

5.4 Résultats en dimension 8

5.4.1 Données avec trois classes bien séparées

Cet ensemble de données comporte 285 points répartis en trois classes bien séparées. Nous obtenons les résultats suivants :

3 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	3	3	3	3	3	3
Saut maximum	+	3	3	3	3	3	3	3	3
Moyenne	+	3	3	3	3	3	3	3	3
Ward	+	3	3	3	3	3	3	3	3

TAB. 5.15 – Résultats obtenus pour l'ensemble de données avec trois classes bien séparées en dimension 8

5.4.2 Données avec deux classes bien séparées

Cet ensemble de données se compose de 200 points répartis en deux classes. Voici les résultats :

2 classes bien séparées		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	2	2
Moyenne	+	2	2	2	2	2	2	2	2
Ward	+	2	2	2	2	2	2	2	2

TAB. 5.16 – Résultats obtenus pour l'ensemble de données avec deux classes bien séparées en dimension 8

5.4.3 Données avec deux classes parallèles

Il s'agit d'un ensemble de données contenant 250 points répartis en deux classes qui sont parallèles. Voici les résultats :

2 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	16	1	1	16	1
Saut maximum	-	1	1	2	12 ou 7	16	10	12	8
Moyenne	-	1	1	2	13 ou 8	1	17	13	8
Ward	-	1	1	2	12 ou 7	15	10	12	7

TAB. 5.17 – Résultats obtenus pour l'ensemble de données avec deux classes parallèles en dimension 8

5.4.4 Données avec trois classes parallèles

Cet ensemble se compose de 375 points répartis en trois classes parallèles. Nous obtenons les résultats suivants :

3 classes parallèles		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	3	3	2	16	16	2	16	1
Saut maximum	-	1	1	5	17 ou 14	1	2	17	13
Moyenne	-	1	1	4	19 ou 14	1	2	19	14
Ward	-	1	1	4	17 ou 14	1	2	17	12

TAB. 5.18 – Résultats obtenus pour l'ensemble de données avec trois classes parallèles en dimension 8

5.4.5 Données "gros-petit" éloignés

Cet ensemble de données contient 260 points répartis en deux classes d'effectifs différents qui sont éloignées. La première compte 200 points et la seconde 60. Nous obtenons les résultats suivants :

"gros-petit" éloignés		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	+	2	2	2	2	2	2	2	2
Saut maximum	+	2	2	2	2	2	2	16	2
Moyenne	+	2	2	2	2	2	2	18	2
Ward	+	2	2	2	2	2	2	15	2

TAB. 5.19 – Résultats obtenus pour l'ensemble de données "gros-petit" éloignés en dimension 8

5.4.6 Données sans structure

Cet ensemble de données contient de 400 points. Nous obtenons les résultats suivants :

Sans structure	M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	1	1	2	1	1	1	15	1
Saut maximum	1	1	6	1	1	2	21	1
Moyenne	1	1	6	1	1	4	18	1
Ward	1	1	2	1	1	2	20	1

TAB. 5.20 – Résultats obtenus pour l'ensemble de données sans structure en dimension 8

5.4.7 Données avec un pont entre les classes

Cet ensemble de données se compose de 249 points répartis en deux classes contenant chacune 105 points et reliées entre elles par un pont de 39 points. Nous obtenons les résultats suivants :

Pont		M1	M2	m1	m2	m3	m4	m5	m6
Saut minimum	-	1	1	3	1	1	1	1	1
Saut maximum	+	2	2	2	15	3	2	15	2
Moyenne	+	2	2 ou 3	2	4	2	2	3	2
Ward	+	2	2 ou 3	2	16	3	2	16	2

TAB. 5.21 – Résultats obtenus pour l'ensemble de données avec un pont entre les classes en dimension 8

5.5 Synthèse des résultats

Dans cette partie, nous allons passer en revue chaque ensemble de données afin de faire quelques commentaires sur les résultats que nous avons obtenus en dimensions 6, 7 et 8. Commençons par **les données bien séparées**. Nous constatons que pour ce type d'ensembles, aucune méthode de détermination du nombre de classes ne semble avoir des difficultés. En effet, les huit méthodes que nous avons utilisées, déterminent toutes le nombre correct de classes qui sont naturelles. Ceci a été constaté pour des ensembles de données dont le nombre de classes était de deux ou de trois.

Passons à présent aux données avec **deux classes parallèles**. Nous constatons que comme dans les dimensions inférieures, seule la méthode du saut mini-

mum détecte la véritable structure des données. Associées à cette classification, les méthodes de détermination du nombre de classes basées sur le critère des hypervolumes et la méthode de Calinski et Harabasz déterminent le nombre correct de classes. Remarquons également qu'en dimensions 6 et 7, une quatrième méthode fournit un résultat correct, il s'agit de la méthode de Duda et Hart classée deuxième par Milligan et Cooper. En dimension 7, la méthode de Beale détermine aussi le bon nombre de classes. Notons que les trois autres méthodes de classification que nous avons utilisées, ne retrouvent pas les classes naturelles. Appliquées à ces méthodes, M1 et M2 concluent à l'absence de structure.

Ajouter une classe aux données précédentes afin d'obtenir un ensemble de données avec **trois classes parallèles** semble perturber les méthodes de détermination du nombre de classes de Milligan et Cooper. En effet, aucune de ces méthodes n'est capable de fournir un résultat correct pour ce genre de données. Les méthodes basées sur le critère des hypervolumes, quant à elles, déterminent le bon nombre de classes quand elles sont associées à une méthode de classification retrouvant les classes naturelles. Dans la situation contraire, elles concluent à l'absence de structure.

Par ailleurs, pour les ensembles de **données d'effectifs fortement différents**, nous remarquons que quand les classes sont éloignées l'une de l'autre, toutes les méthodes de détermination du nombre de classes sauf celle de Beale retrouvent le nombre correct de classes.

Pour l'ensemble de **données sans structure**, nous remarquons qu'en plus de la méthode de Calinski et Harabasz qui a toujours eu des difficultés pour ce genre de données, la méthode de Beale en éprouve également ici.

Enfin, terminons par les **données avec un pont entre les classes**. Nous constatons que la quasi-totalité des méthodes de détermination conclut à l'absence de structure quand elles sont associées à la méthode du saut minimum. Comme précédemment, ce résultat est lié à l'effet de chaînage. Quand elles sont appliquées aux trois autres méthodes de classification, les méthodes m1, m3, m4 et m6 déterminent qu'il y a deux ou trois classes dans les données.

Chapitre 6

Conclusions

L'objet de ce mémoire porte sur l'analyse du comportement des méthodes de détermination du nombre de classes dans des espaces multidimensionnels. Il essaie plus précisément de mettre en évidence la contribution spécifique du critère des hypervolumes à la détermination du nombre de classes. Pour ce faire, nous avons comparé deux méthodes de détermination du nombre de classes basées sur ce critère aux six meilleures méthodes de détermination du nombre de classes de Milligan et Cooper. Cette comparaison étant limitée dans les études existantes à des espaces de données où le nombre de caractéristiques observées sur chaque individu était de 2 ou de 3, nous l'avons étendue à des ensembles dont la dimension variait entre 4 et 8. Ceci nous a permis de confirmer plusieurs conclusions qui avaient été tirées sur des ensembles de données de dimension inférieure.

Mais avant d'exposer les conclusions relatives aux méthodes de détermination du nombre de classes, il est impératif, lorsqu'on recherche à déterminer le nombre correct de classes naturelles, d'appliquer ces méthodes à des hiérarchies de partitions contenant les classes naturelles. En effet, pour les ensembles de données traités, nous avons constaté que peu de méthodes déterminaient le nombre correct de classes lorsqu'elles étaient associées à une méthode de classification ne retrouvant pas les classes naturelles. Par ailleurs, même si dans ce cas, le nombre correct de classes était retrouvé, les classes proposées ne reflétaient pas la véritable structure des données. Il apparaît dès lors capital de tenir compte des biais des méthodes de classification dans le problème de détermination du nombre de classes.

Nous avons tout d'abord constaté qu'à quelques exceptions près, la dimension de l'espace n'a pas influencé l'obtention des résultats. C'est la raison pour laquelle, nous n'y ferons pas référence dans la présentation des conclusions tirées

lors de l'application des méthodes de détermination du nombre de classes à différents ensembles de données.

Pour les deux méthodes basées sur **le critère des hypervolumes**, l'application de la méthode du coude et de la méthode basée sur un test du quotient de vraisemblance à la suite d'une méthode de classification retrouvant les classes naturelles a déterminé le nombre correct de classes pour chaque type d'ensembles de données considérés dans ce mémoire. Par contre, lorsque ces deux méthodes ont été appliquées à une hiérarchie qui ne contenait pas les classes naturelles, elles ont toujours conclu à l'absence de structure. Par conséquent, les deux méthodes de détermination du nombre de classes basées sur le critère des hypervolumes fournissent d'excellents résultats lorsqu'elles sont associées à des méthodes de classification autres que la méthode de classification des hypervolumes. Une conclusion similaire avait déjà été tirée lors de l'application de ces deux méthodes à des ensembles de données de dimension 2. Ceci montre donc qu'il est possible de déterminer le nombre correct de classes naturelles sans devoir utiliser la méthode de classification des hypervolumes dont la complexité augmente avec la dimension de l'espace.

Les **six meilleures méthodes** de détermination du nombre de classes de **Milligan et Cooper** n'ont pas donné d'aussi bons résultats. En effet, aucune de ces six méthodes lorsqu'elle était appliquée à la suite d'une méthode de classification retrouvant les classes naturelles, n'a été capable de déterminer le nombre correct de classes pour tous les types d'ensembles de données traités. Le tableau 6.1 présente les performances et les défaillances des méthodes de Milligan et Cooper à déterminer le nombre correct de classes en fonction du type d'ensembles de données.

Chaque colonne de ce tableau correspond à une méthode de détermination du nombre de classes tandis que chaque ligne se rapporte à un jeu de données. En outre, lorsqu'une méthode de détermination du nombre de classes associée à une classification contenant les classes naturelles fournissait un résultat correct pour un type d'ensembles de données, le caractère "X" à l'intersection de la ligne et de la colonne correspondant respectivement à ce jeu de données et à cette méthode a été employé. Par contre, lorsque le nombre de classes déterminé était incorrect, le symbole "-" a été utilisé.

	Méthode de Calinski et Harabasz	Méthode de Duda et Hart	Méthode du C-index	Méthode Gamma	Méthode de Beale	Méthode CCC
Données avec 3 classes séparées	X	X	X	X	X	X
Données avec 2 classes séparées	X	X	X	X	X	X
Données avec 2 classes parallèles	X	*	-	-	-	-
Données avec 3 classes parallèles	-	-	-	-	-	-
Données "gros-petit" éloignés	X	X	X	X	-	X
Données "gros-petit" proches	X	X	X	-	-	-
Données sans structure	-	X	X	X	**	X
Données avec un pont	X	-	X	X	-	X

TAB. 6.1 - Synthèse des résultats pour les méthodes de Milligan et Cooper

Par ailleurs, le symbole "*" à l'intersection de la ligne "données avec deux classes parallèles" et de la colonne "méthode de Duda et Hart" désigne le fait que pour cette dernière, nous avons eu des résultats qui variaient avec la dimension de l'espace. En effet, cette méthode détermine le nombre correct de classes en dimensions 6 et 7 alors que dans les autres dimensions, elle fournit une solution différente. L'interprétation du symbole "**" à l'intersection de la ligne "données sans structure" et "méthode de Beale" est similaire : cette méthode ne détecte plus l'absence de structure à partir de la dimension 6.

Le tableau 6.1 montre que la majorité des six méthodes éprouve des difficultés à déterminer le nombre correct de classes lorsque les données ont une structure plus particulière que celles avec des classes bien séparées. Nous faisons référence notamment aux données avec des classes parallèles et aux données sans structure. Rappelons toutefois que ces types d'ensembles de données n'ont pas été utilisés dans l'étude de Milligan et Cooper.

Les résultats relatifs aux méthodes de Milligan et Cooper ont en outre permis de montrer que la hiérarchie établie par ces derniers se confirme relativement bien pour les ensembles de données utilisés, si ce n'est peut-être la place attribuée à la méthode de Beale. Mais, il ne faut pas négliger que ces derniers ne sont pas très nombreux. C'est pourquoi, nous pensons qu'il serait intéressant d'appliquer ces méthodes à un plus grand nombre d'ensembles de données présentant des structures aussi variées que les nôtres. Cela permettrait sans doute d'identifier avec plus de certitude les biais de ces six méthodes.

Enfin, l'application des six méthodes de détermination du nombre de classes et de celles basées sur le critère des hypervolumes aux iris de Fisher a montré toute la difficulté de travailler avec des données réelles. En outre, nous pensons qu'il serait intéressant d'appliquer les huit méthodes de détermination du nombre de classes considérées à d'autres ensembles de données issues de la réalité.

Bibliographie

[1] ANDRE, P. (1997), *Détermination du nombre de classes en classification automatique*, mémoire de fin d'études, FUNDP, Namur.

[2] BAKER, F.B. et HUBERT, L.J. (1975), Measuring the power of hierarchical cluster analysis, *Journal of the American Statistical Association*, 70, 31-38.

[3] BEALE, E.M.L. (1969), Cluster analysis, London : Scientific Control Systems.

[4] CALINSKI, R.B. et HARABASZ, J. (1974), A dendrite method for cluster analysis, *Communications in Statistics*, 3, 1-27.

[5] CHANDON, J.L. et PINSON, S. (1981), *Analyse typologique : théorie et applications*, Masson, Paris.

[6] DALRYMPLE-ALFORD, E.C. (1970), The measurement of clustering in free recall, *Psychological Bulletin*, 75, 32-34.

[7] DESCHAMPS, J.-F. (1998), *Etude comparative de neuf procédures de détermination du nombre de classes en classification automatique*, mémoire de fin d'études, FUNDP, Namur.

[8] DUDA, R.O. et HART, P.E. (1973), *Pattern classification and scene analysis*, Wiley-Interscience, New York.

[9] EVERITT, B. (1980), *Cluster analysis*, Halsted press, London.

[10] EVERITT, B. (1993), *Cluster analysis*, Arnold, London.

[11] GORDON, A.D. (1998), How many clusters? An investigation of five procedures for detecting nested cluster structure, *Data Science, Classification,*

and Related Methods, 109-116, Springer Verlag, Berlin.

[12] HARDY, A. (1983), *Statistique et classification automatique. Un modèle - Un nouveau critère - Des algorithmes - Des applications*, Thèse de doctorat, FUNDP, Namur.

[13] HARDY, A. (1996), On the number of clusters, *Computational Statistics and Data Analysis*, 23, 83-96.

[14] HUBERT, L.J. et LEVIN, J.R. (1976), A general statistical framework for assessing categorical clustering in free recall, *Psychological Bulletin*, 83, 1072-1080.

[15] MILLIGAN, G.W. et COOPER, M.C. (1985), An examination of procedures for determining the number of clusters in a data set, *Psychometrika*, 50, 159-179.

[16] MOORE, M. (1984), On the estimation of a convex set, *The Annals of Statistics*, 12, 3, 1090-1099.

[17] RASSON, J.P. (1979), Estimation de formes convexes du plan, *Statistique et Analyse des Données*, 1, 31-46.

[18] RIPLEY, B.D. et RASSON, J.P. (1977), Finding the edge of a Poisson forest, *Journal of Applied Probability*, 14, 483-491.

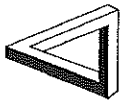
[19] SARLE, W.S. (1983), Cubic Clustering Criterion, Technical Report : A-108, SAS Institute Inc..

[20] (1990), *SAS/STAT User's Guide*, SAS Institute Inc., Cary, NC, USA .

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Contribution spécifique du critère des hypervolumes à la détermination du nombre de classes dans des espaces multidimensionnels : annexes



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

Promoteur : A. Hardy

Florence PIRARD

Année Académique 1999-2000

Table des matières

Annexe A

Programme pour les méthodes basées sur le critère des hypervolumes

```
      subroutine testhyp (k,dim,n,r)

c     Testhyp calcule la valeur du quotient de vraisemblance sur base des
c     partitions en k et k-1 classes contenues respectivement dans les
c     fichiers partk.dat et partj.dat.

      IMPLICIT NONE

c     Déclaration et description des argumemts de testhyp
c     ---------------------------------------------------

      INTEGER k,dim,n
      DOUBLE PRECISION r

c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observées sur chaque individu
c     n : Nombre d'individus
c     r : Valeur du quotient de vraisemblance


c     Déclaration et description des variables utilisées dans testhyp
c     ---------------------------------------------------------------

      DOUBLE PRECISION v,w

c     v : Valeur du critère des hypervolumes pour la partition en k-1 classes
c     w : Valeur du critère des hypervolumes pour la partition en k classes
```

```
c     fichier contenant la partition en k classe (la forme que doit avoir ce
c     fichier est expliquée dans la sous-routine crithyp)

      open (unit=20, file='partk.dat', status='old')

c     fichier contenant la partition en k-1 classe (la forme que doit avoir ce
c     fichier est expliquée dans la sous-routine crithyp)

      open (unit=30, file='partj.dat', status='old')

c     Calcul du critère des hypervolumes en k et k-1 classes

      call crithyp(k,dim,n,20,w)
      call crithyp(k-1,dim,n,30,v)

      write(*,*)'Valeur du critère des hypervolumes en k-1 classes =',v
      write(*,*)'Valeur du critère des hypervolumes en k classes =',w

c     Calcul de la valeur du quotient de vraisemblance

      r=w/v

      close(20)
      close(30)

      end

c     ************************************************
      program test

c     Programme principal


      DOUBLE PRECISION r

      call testhyp (3,5,64,r)
      write(*,*) 'la valeur du quotient est= ',r
      end

c     ************************************************

      SUBROUTINE crithyp(k,dim,n,u,w)
```

```
c     Crithyp calcule la valeur du critère des hypervolumes, à savoir la somme
c     des mesures de Lebesgues des enveloppes convexes des k classes. Pour
c     calculer cette valeur, on dispose d'une partition en k classes obtenue
c     par SAS.
c
c     Le fichier contenant cette partition doit être de la forme suivante :

c     _NAME_ X Y CLUSTER CLUSNAME
c     OB1 1 3 1 CL6
c     OB3 2 2 1 CL6
c     OB2 1 1 1 CL6
c     OB4 3 3 1 CL6
c     OB5 3 1 1 CL6
c     OB6 10 10 2 CL2
c     OB8 11 11 2 CL2
c     OB7 12 10 2 CL2
c     OB9 10 12 2 CL2
c     OB10 12 12 2 CL2

c     REMARQUE
c     --------
c     La colonne _NAME_ correspond au nom donné aux individus (OBi =
c     le nom de l'individu qui occupe la ième ligne du fichier de données).
c     Les colonnes X et Y correspondent aux variables qui ont été observées.
c     La colonne CLUSTER correspond au numéro de la classe où l'objet a été
c     classé.
c     La colonne CLUSNAME correspond au nom de la classe.



c     Cette partition est obtenue en utilisant les procédures suivantes dans
c     SAS:

c      data ond;
c      infile 'carre.dat';
c      input var1-var2;
c      run;
c      proc cluster ccc
c       data=ond
c       method=single
c      outtree=arbre;
c      run;

c      proc tree
c      data=arbre
```

```
c      noprint
c      ncl=2
c      out=o;
c     copy var1-var2;
c      run;

c      proc sort
c      data=o
c      out= tri;
c      by cluster;
c      run;

c      proc print
c      data=tri;
c      var cluster;
c      run;


c     Remarquons que ce fichier est trié par classe.
c                                    ----

      IMPLICIT NONE

c     Déclaration et description des argumemts de crithyp
c     ---------------------------------------------------

      INTEGER k,dim,n,u
      DOUBLE PRECISION w

c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observées sur chaque individu
c     n : Nombre d'individus
c     u : Numéro de l'unité associée au fichier contenant les données
c     w : Somme des hypervolumes des enveloppes convexes des k classes


c     Déclaration et description des variables utilisées dans crithyp
c     ---------------------------------------------------------------

      INTEGER x(dim,n),cl(n),nb(0:k),c(dimmax,npmax)
      INTEGER i,j,z,f
      DOUBLE PRECISION volume
      CHARACTER*100 titre
      CHARACTER*8 nom(n)
      CHARACTER*4 clnom(n)
```

```
c     titre : Contient la première ligne du fichier des données
c     nom(n) : Vecteur contenant le nom associé à chaque individu (nom(i)
c              contient le nom associé à ième individu du fichier)
c     x(dim,n) : Matrice contenant les caractéristiques de tous les individus
c                ( x(j,i) contient la caractéristique j du ième individu )
c     cl(n) : Vecteur qui contient le numéro de la classe des individus
c             ( cl(i) est le numéro de la classe du ième individu )
c     clnom(n) : Vecteur contenant le nom de la classe associé à chaque
c                individu (clnom(i) contient le nom de la classe du ième
c                individu )
c     nb(0:k) : Vecteur qui contient l'indice du dernier individu de chaque
c               classe ( nb(i) est l'indice du dernier individu de la ième
c               classe )
c     c(dimmax,npmax) : Matrice qui contient les caractéristiques des
c                       individus appartenant à la même classe
c     z : contient le nombre d'éléments de la ième classe à la ième itération
c     f : contient la valeur de nb(i-1) a la ième itération
c     volume : contient l'hypervolume de la ième classe à la ième itération
c     i,j : indice de boucle


c     Crithyp fait appel à la sous-routine volclasse (k,dim,n,z,f,x,c,volume)
c     qui calcule l'hypervolume de l'enveloppe convexe de chacune des classes.

c     Initialisation de la valeur du critère

      w=0.

c     Lecture du fichier contenant les données

      read(u,100) titre
      do  i=1,n
         read(u,*)nom(i),(x(j,i), j=1,dim),cl(i),clnom(i)
      enddo
 100  format (a100)



c     Evaluation du vecteur nb

      j=1
      nb(0)=0
      do i=1,k
 200     continue
```

```
            if ((cl(j).eq.i).and.(j.le.n))then
               j=j+1
               goto 200
            endif
            nb(i)=j-1
         enddo

c        Calcul de l'hypervolume de l'enveloppe convexe de chaque classe.

         do i=1,k
            z=nb(i)-nb(i-1)
            f=nb(i-1)
            call volclasse (k,dim,n,z,f,x,c,volume)
            w=w+volume
         enddo

         end
c        ******************************************************
         SUBROUTINE volclasse (k,dim,n,z,f,x,c,volume)

c        volclasse calcule l'hypervolume de l'enveloppe convexe de la ième
c        classe.

         IMPLICIT NONE

         include 'CONSTMAX.DEF'
         include 'FATAL_LIMITS.DEF'
c        Ces deux fichiers se trouvent à la fin du code

c        Déclaration et description des argumemts de volclasse
c        -----------------------------------------------------

         INTEGER k,dim,n,z,f,x(dim,n),c(dimmax,z)
         DOUBLE PRECISION volume

c        k : Nombre de classes constituant la partition
c        dim : Nombre de variables observées sur chaque individu
c        n : Nombre d'individus
c        z : contient le nombre d'éléments de la ième classe
c        f : contient la valeur de nb(i-1)
c        x(dim,n) : Matrice contenant les caractéristiques de tous les individus
c                   ( x(j,i) contient la caractéristique j du ième individu
c        c(dimmax,npmax) : Matrice qui contient les caractéristiques des
c                          individus appartenant à la même classe
c        volume : contient l'hypervolume
```

```
c     Déclaration et description des variables utilisées dans volclasse
c     ------------------------------------------------------------------

      INTEGER l,j

c     l,j : indice de boucles

c     volclasse fait appel à la sous-routine convexe. La déclaration de ses
c     arguments est :

      INTEGER          face    (dimmax,nfmax)
      INTEGER hyper1 (3,nhmax)
      INTEGER face1  (nfmax)
      INTEGER nh,nf,h
      DOUBLE PRECISION surface
      DOUBLE PRECISION hyperplan (dimmax,nhmax)
      DOUBLE PRECISION vectra    (dimmax)


      do l=1,dim
         do j=1,z
            c(l,j)=x(l,f+j)
         enddo
      enddo
      call convexe (c,hyperplan,vectra,face,hyper1,face1,
     +     dim,z,nh,nf,volume,h,surface,.false.)
      end



c     ***********************************************************************
      SUBROUTINE CONVEXE (POINT,HYPERPLAN,VECTRA,FACE,HYPER1,
     +                    FACE1,DIM,NP,NH,NF,VOLUME,H,SURFACE,COND)
C
C     ***********************************************************************
C
C     Cette sous-routine nous a été fournie. Elle a été écrite par
c     L. Waerenburgh.
C
C     BUT DE LA SOUS-ROUTINE :
C     ------------------------
C
C          Calculer l'enveloppe convexe d'un ensemble de points d'un
```

```
C espace de dimension DIM. La methode utilisee n'est pas recursive.
C
C ***********************************************************************
C
      IMPLICIT NONE
C
      INCLUDE 'CONSTMAX.DEF'
      INCLUDE 'FATAL_LIMITS.DEF'
C
C
C SPECIFICATION DES ARGUMENTS DE LA SOUS-ROUTINE :
C ----------------------------------------------
C
INTEGER          POINT     (DIMMAX,*)
DOUBLE PRECISION HYPERPLAN (DIMMAX,*)
DOUBLE PRECISION VECTRA    (DIMMAX)
INTEGER          FACE      (DIMMAX,*)
C
INTEGER HYPER1 (3,*)
INTEGER FACE1  (*)
C
INTEGER          DIM
INTEGER          NP,NH,NF
INTEGER          H
DOUBLE PRECISION VOLUME
DOUBLE PRECISION SURFACE
LOGICAL          COND
C
C
C
C DESCRIPTION DES ARGUMENTS DE LA ROUTINE
C --------------------------------------
C
C       DIMMAX : variable entiere contenant la dimension de declaration des
C          tableaux dans le programme appelant. Cela correspond à la
C          dimension maximale du probleme.
C       NPMAX : nombre maximum de points.
C       NCBMAX : nombre maximum de clusters.
C       NHMAX : nombre maximum d'hyperplans.
C       NFMAX : nombre maximum de faces.
C POINT : tableau de dimension 2 d'entiers. Il contient les coordonnees
C         des differents points dont il faut chercher l'enveloppe
C         convexe. Les points (au nombre de NP) appartiennent à un
C               espace de dimension DIM. Ils sont donc caractérisés par DIM
C               entiers (DIM <=DIMMAX).
```

```
C HYPERPLAN : tableau de dimension 2 de reels. Il contient les
C             coefficients des equations des hyperplans.
C             A la sortie de la routine, il y aura NH hyperplans.
C             Les equations seront de la forme :
C                          somme   (alpha  * x ) = 1 ou 0
C                         i=1,dim        i    i
C             Un hyperplan, dans un espace de dimension DIM, est donc
C             caracterise par dim coefficients 'alpha' et un terme
C             independant. Le terme independant se trouvera, a la
C             sortie, dans la premiere ligne du tableau HYPER1.
C             Les coefficients 'alpha' seront dans le tableau HYPERPLAN
C             Les 'x' correspondent aux coordonnees d'un point. LE terme
C             independant peut etre 0 ou 1 suivant que l'hyperplan
C             passe ou non par l'origine (0,...,0).
C             HYPERPLAN(I,J) est le ieme coeffic. de l'hyperplan
C             numero j.
C       VECTRA : tableau de dimension 1 de reels. Il s'agit d'un vecteur de
C          vecteur de translation utilise pour que le point de
C          coordonnees (0,0,...,0) soit interieur au polyedre de base.
C          Ceci facilite la recherche des equations des hyperplans.
C          Si COND est 'true', les equations des hyperplans seront
C                relatives a ce vecteur. Sinon, elles seront absolues (i.e.
C                relatives a la vraie origine (0,...,0)).
C       FACE : tableau de dimension 2 d'entiers. Il contient les references
C        des points composant une face. Cette reference correspond au
C        tableau POINT. Il faut faire tres attention au fait que pour
C              nous, une face est un ensemble de DIM points differents.
C              Par exemple :
C                     - en dimension 1 : face = un point
C                     - en dimension 2 : face = un segment
C                     - en dimension 3 : face = un triangle
C                     - en dimension 4 : face = un tetraedre
C                     - ...
C        De ce fait, un meme hyperplan peut contenir plusieurs faces.
C              FACE(I,J) est le numero du ieme point de la jeme face.
C       HYPER1 : tableau de dimension 2 d'entiers.
C          HYPER1(1,J) = statuts de l'hyperplan j.
C                      = 0 : l'hyperplan est interne au polyedre courant.
C                      = 1 : l'hyperplan est transitoire.
C                      = 2 : l'hyperplan est final.
C          HYPER1(2,J) = nbre de faces de l'hyperplan j
C  HYPER1(3,j) = no de la 1er face de l'hyperplan j
C          A la sortie de la sous-routine, seuls les hyperplans
C          finaux sont conserves. Des lors, la premiere ligne de
C          HYPER1 n'a plus de raison d'etre. On y met le terme in-
```

```
C          dependant de l'equation de l'hyperplan.
C       FACE1 : tableau de dimension 1 d'entiers contenant la reference
C         d'une face appartenant au meme hyperplan. S'il n'y en a pas
C         d'autre, sa valeur est 0.
C         FACE1(I) = no de la face suivante appartenant au meme hyperplan
C                    que la face i
C       DIM : variable entiere contenant la dimension du probleme.
C NP : variable entiere contenant le nombre de points dont il faut
C      chercher l'enveloppe convexe.
C NH : variable entiere contenant le nombre courant d'hyperplans.
C           plans. A la sortie, cette variable contiendra le nombre
C           d'hyperplans finaux.
C       NF : variable entiere contenant le nombre courant de faces. A la
C      sortie, cette variable contiendra le nombre de faces inclues dans
C      les hyperplans finaux.
C       VOLUME : variable DOUBLE PRECISION contenant les hypervolumes des
C                differents convexes.
C SURFACE: variable DOUBLE PRECISION contenant la surface du polyhedre
C       COND : variable logique etant a 'true' si les equations des
C        hyperplans, a la sortie, sont relatives à VECTRA et
C        'false' si elles sont absolues.
C
C
C DECLARATION DES CONSTANTES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------
C
DOUBLE PRECISION PREC1,PREC2,PREC
C
PARAMETER (PREC1=0.,PREC2=0.,PREC=1D-10)
C
C
C LISTE DES CONSTANTES UTILISEES DANS LA SOUS-ROUTINE :
C -------------------------------------------------
C
C       NHEMAX : constante entiere contenant le nombre maximal d'hyperplans
C          vis-a-vis desquelles un meme point peut etre exterieur.
C       NFEMAX : constante entiere contenant le nombre maximal de faces
C          vis-a-vis desquelles un meme point peut etre exterieur.
C NNFMAX : constante entiere contenant le nombre maximal de nouvelles
C          faces pouvant etre engendrees par un meme point.
C PREC1 : constante DOUBLE PRECISION contenant la precision desiree
C         lors du calcul des equations des hyperplans.
C PREC2 : constante DOUBLE PRECISION contenant la precision desiree
C         lors de la comparaison des equations des hyperplans.
C       LIMIT_RATIO : constante entiere contenant la proportion de points
```

```
C       entrant dans la construction de l'enveloppe convexe
C                     Cette constante peut donc etre utilisee pour obtenir
C                     une approximation de l'enveloppe convexe par
C       l'interieur. Elle n'est utilisee ici qu'a partir de
C       DIM = 6 et peut valoir par exemple 90 ou 95.
C
C
C DECLARATION DES VARIABLES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------
C
INTEGER NPS,NPI
INTEGER NHT,NHI,NHF
C
INTEGER NFE,     FACEXT (NFEMAX)
INTEGER NNF,     NOUFAC (DIMMAX,NNFMAX)
INTEGER NH1,NH2,LIHYIN(NHEMAX),LIHYEX(NHEMAX)
C
INTEGER INITIA (DIMMAX)
INTEGER HT,PT
C
INTEGER          X(DIMMAX),Y(DIMMAX),TERIND
INTEGER          POINT1(NPMAX)
DOUBLE PRECISION POINT2(DIMMAX,NPMAX)
DOUBLE PRECISION BASE(DIMMAX,DIMMAX)
DOUBLE PRECISION A(DIMMAX,DIMMAX),B(DIMMAX)
C
INTEGER          I,J,K,L,M
C INTEGER          I1,I2,I3,I4,I5,I6
DOUBLE PRECISION SOMME,DET(2),AUX
double precision cg(DIMMAX),vol,cgaux(DIMMAX)
LOGICAL          EGAL

INTEGER LIMIT_RATIO
PARAMETER (LIMIT_RATIO = 90)
INTEGER NP_LIMIT_RATIO
C
C
C LISTE DES VARIABLES UTILISEES DANS LA SOUS-ROUTINE :
C ------------------------------------------------
C
C       POINT1 : tableau de dimension 1 d'entiers. Il contient les statuts
C          associes a chaque point. On considere qu'il y a trois
C          statuts possibles durant l'execution de la sous-routine
C                    . -1 : le point n'a pas encore ete traite.
C                    .  0 : le point est interne au polyedre courant.
```

```
C                    . +1 : le point est sommet du polyedre courant.
C          A la sortie de la sous-routine, seuls les deux derniers
C          statuts restent actifs (i.e. on associe a chaque point
C          un des deux derniers statuts).
C NPS : variable entiere contenant le nombre de points consideres
C       comme etant des sommets (ou pseudo-sommets).
C       NPI : variable entiere contenant le nombre de points consideres
C       comme etant internes.
C NHT : variable entiere contenant le nombre d'hyperplans, parmi
C       les NH hyperplans, consideres comme etant transitoires.
C NHI : variable entiere contenant le nombre d'hyperplans, parmi
C            les NH hyperplans, consideres comme etant internes.
C NHF : variable entiere contenant le nombre d'hyperplans, parmi
C             les NH hyperplans, consideres comme etant finaux.
C NFE : variable entiere contenant le nombre de faces pour lesquelles
C       le point PT est exterieur.
C FACEXT : tableau de dimension 1 d'entiers. Il contient les references
C          des faces pour lesquelles le point PT est exterieur.
C          Les references correspondent au tableau FACE. Ce tableau peut
C          contenir au maximum NFEMAX de references.
C       NNF : variable entiere contenant le nombre de nouvelles faces
C       formees a partir du point PT de de l'hyperplan HT.
C       NOUFAC : tableau de dimension 2 d'entiers. Il contient les references
C          des points composant les nouvelles faces formees a partir
C          du point PT et de l'hyperplan HT. Les references
C          correspondent au tableau POINT. Ce tableau peut
C          contenir au maximum NNFMAX faces, decrites par maximum
C          DIMMAX points.
C NH1 : variable entiere contenant le nombre d'hyperplans pour
C       lesquelles le point PT devra etre interieur (pour etre
C       inclus dans le convexe).
C       NH2 : variable entiere contenant le nombre d'hyperplans pour
C       lesquelles le point PT devra etre exterieur (pour etre
C       inclus dans le convexe).
C LIHYIN : tableau de dimension 1 d'entiers. Il contient les references
C          des hyperplans pour lesquels le point PT est interieur.
C LIHYEX : tableau de dimension 1 d'entiers. Il contient les references
C          des hyperplans pour lesquels le point PT est exterieur.
C INITIA : tableau de dimension 1 d'entiers. Il est utilise dans la
C          construction du polyedre de base. Il contient les references
C          des differents points composant le polyedre de base.
C          Les references correspondent au tableau POINT.
C HT : variable entiere contenant l'indice du premier hyperplan
C      transitoire non encore traite (hyperplan transitoire courant).
C PT : variable entiere contenant l'indice du point le plus exterieur
```

```
C      a l'hyperplan HT.
C X : tableau de dimension 1 d'entiers. Il est utilise dans la recherche
C     des equations des hyperplans, et plus particulierement dans la
C     resolution du systeme d'equations. Il est egalement utilise lors
C           de la recherche du volume.
C Y : tableau de dimension 1 d'entiers. Il est utilise dans
C     la recherche des equations des hyperplans, et plus
C     particulierement dans la resolution du systeme d'equations.
C TERIND : variable entiere contenant le terme independant de l'
C          equation d'un hyperplan. Ce terme peut prendre deux
C          valeurs possibles : 0 si l'hyperplan passe par l'origine
C          et 1 sinon.
C POINT2 : tableau de dimension 2 de reels. Il contient les
C          cooordonnees des differents points. Il peut donc contenir au
C                maximum NPMAX points d'un espace de dimension maxim
C BASE : tableau de dimension 2 de reels. Il est utilise pour la
C              determination du polyedre de base, plus particulierement pour
C              la recherche des differents sommets constituant le polyedre de
C              base.
C A : tableau de dimension 2 de reels. Il est utilise dans la recherche
C           des equations des hyperplans, et ce plus particulierement dans la
C           resolution du systeme d'equations.
C B : tableau de dimension 1 de reels. Il est utilise dans la recherche
C           des equations des hyperplans, et ce plus particulierement dans la
C           resolution du systeme d'equations.
C I,J,K,L,M : variables entieres utilisees comme indices de boucle.
C SOMME : variable DOUBLE PRECISION contenant le resultat de la
C         fonction SOM.
C DET : tableau de dimension 1 de reels. Il est utilise dans le calcul
C             du determinant.
C AUX : variable DOUBLE PRECISION auxilliaire.
C       EGAL : variable logique utilisee lors de la comparaison des
C        equations des hyperplans
C
C
C DECLARATION DES FONCTIONS UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------
C
DOUBLE PRECISION SOM
INTEGER          REPOEL,REPOEX
LOGICAL          INT,EXT
C
EXTERNAL  SOM,REPOEL,REPOEX,INT,EXT
INTRINSIC ABS
C
```

```
C
C LISTE DES FONCTIONS ET ROUTINES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------------
C
C       SOM (DIM,POINT,HYPERPLAN) : fonction DOUBLE PRECISION qui
C     calcule le produit scalaire du vecteur POINT avec le
C     vecteur HYPERPLAN, les deux vecteurs etant de dimension DIM.
C       REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,TERIND) : fonction
C           entiere qui renvoie l'indice du point le plus eloigne de
C           l'hyperplan specifie. La recherche du point ne s'effectue que
C           sur les points exterieurs au polyedre courant.
C REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN) : fonction entiere
C     qui renvoie l'indice du point le plus exterieur
C     a l'hyperplan specifie. La recherche du point ne s'
C     effectue que sur les points exterieurs au polyedre courant.
C INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est interieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est exterieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,DIM,NNF,NOUFAC) :
C     sous-routine qui recherche les nouvelles faces engendrees par le
C           point PT et par les NFE faces dont les indices se trouvent dans
C           le tableau FACEXT. Cette routine exclut les faces internes.
C REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,DIM,TERIND,PREC) :
C     sous-routine qui recherche l'equation d'un hyperplan.
C     Plus particulierement, elle recherche les coefficients de
C     l'hyperplan devant contenir les points de la face specifiee
C     Les coefficients obtenus constituent l'hyperplan
C     specifie. La variable TERIND contient la valeur du terme
C     independant de l'equation de l'hyperplan.
C       PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,HYPER1,FACE1,DIM,DIMMAX) :
C           sous-routine qui purge les hyperplans et les faces et nettoye
C           les tableaux.
C
C ***************************************************************
C *  INITIALISATION :                                           *
C *                                                             *
C *     - initialisation de POINT2, de POINT1 et de VECTRA      *
C ***************************************************************
C BEGIN
C *****
c
       Fatal_Limit_NHMAX = NHMAX-1
```

```
        Fatal_Limit_NFMAX = NFMAX-DIMMAX
        Fatal_Limit_NPMAX = NPMAX



FATAL_LIMIT_NHEMAX = NHEMAX - 1
FATAL_LIMIT_NFEMAX = NFEMAX - 1
FATAL_LIMIT_NNFMAX = NNFMAX - 1
IF (NP .GT. NPMAX) THEN
c     WRITE (MSG_UNIT,*) 'NP=', NP
c     WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
    STOP
ENDIF
c




C -----------------------------------------------
C initialisation de POINT2, de POINT1 et de VECTRA
C -----------------------------------------------
C
if (np .lt. 20) then
c           write (MSG_UNIT,*) ' # points:', np
c     write (MSG_UNIT,*) '**********************'
          do i=1,np
c             write (MSG_UNIT,'(3i6)') ( point (j,i) ,j=1, dim)
          enddo
        endif
C
C On construit une fenetre autour de chaque point, c.a.d qu'on ajoute
C autour de chaque point les sommets d'un cube centre en ce point, et
C dont la longueur du demi-cote vaut h
C
K=NP
c DO I=1,NP
c
c
c   IF (DIM.EQ.1) THEN
c           DO I1=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
```

```
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)        ! facteur h...
c     ENDDO
c   ENDIF
c   IF (DIM.EQ.2) THEN
c     DO I1=0,1
c     DO I2=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c        POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c     ENDDO
c     ENDDO
c   ENDIF
c   IF (DIM.EQ.3) THEN
c     DO I1=0,1
c     DO I2=0,1
c           DO I3=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c        POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c        POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c     ENDDO
c     ENDDO
c     ENDDO
c   ENDIF
c     IF (DIM.EQ.4) THEN
c     DO I1=0,1
c     DO I2=0,1
c     DO I3=0,1
c           DO I4=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c        POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c        POINT(3,K)=POINT(3,I)+H*(2*I3-1)
```

```
c         POINT(4,K)=POINT(4,I)+H*(2*I4-1)
c       ENDDO
c       ENDDO
c       ENDDO
c       ENDDO
c     ENDIF
c     IF (DIM.EQ.5) THEN
c       DO I1=0,1
c       DO I2=0,1
c       DO I3=0,1
c       DO I4=0,1
c             DO I5=0,1
c          K=K+1
c          IF (K .GT. NPMAX) THEN
c             WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c             STOP
c          ENDIF
c          POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c          POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c          POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c          POINT(4,K)=POINT(4,I)+H*(2*I4-1)
c          POINT(5,K)=POINT(5,I)+H*(2*I5-1)
c       ENDDO
c       ENDDO
c       ENDDO
c       ENDDO
c       ENDDO
c     ENDIF
c     IF (DIM.EQ.6) THEN
c       DO I1=0,1
c       DO I2=0,1
c       DO I3=0,1
c       DO I4=0,1
c       DO I5=0,1
c             DO I6=0,1
c          K=K+1
c          IF (K .GT. NPMAX) THEN
c             WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c             STOP
c          ENDIF
c          POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c          POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c          POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c          POINT(4,K)=POINT(4,I)+H*(2*I4-1)
c          POINT(5,K)=POINT(5,I)+H*(2*I5-1)
```

```
c         POINT(6,K)=POINT(6,I)+H*(2*I6-1)
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c    ENDIF
c ENDDO
c
c
DO I=1,NP
   DO J=1,DIM
      POINT2 (J,I) = POINT (J,I)
           ENDDO
   POINT1 (I) = -1
    ENDDO



DO I=1,DIM
   VECTRA (I) = 0.
   cg (i) = 0.
    ENDDO
C
C ***********************************************************************
C *  CONSTRUCTION DU POLYEDRE DE BASE :                                 *
C *                                                                     *
C *     - recherche des DIM+1 sommets de base                           *
C *     - tri par ordre croissant des DIM+1 sommets de base             *
C *     - determination du centre de gravite et translation des points  *
C *     - determination des DIM+1 faces de base                         *
C *     - determination des DIM+1 hyperplans de base                    *
C *     - recherche des points internes                                 *
C ***********************************************************************
C
C ---------------------------------
C recherche des DIM+1 sommets de base
C ---------------------------------
C
C initialisation de BASE, de FACE(.,1) et de HYPERPLAN(.,1)
C
DO I=1,DIM
   DO J=1,DIM
      BASE (J,I) = 0.
```

```
   50    ENDDO
   BASE (I,I) = 1.
   FACE (I,1) = I
   HYPERPLAN (I,1) = 1.
   40 ENDDO
TERIND = 1
C
C recherche des DIM sommets formant l'hyperplan de base
C
EGAL = .TRUE.
I = 0
   60 CONTINUE
IF (EGAL.AND.(I.LT.DIM)) THEN
C
C    rechercher le point le plus eloigne de l'hyperplan de base
   PT = REPOEL (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,1),
     +                  TERIND)
C
c    write(MSG_UNIT,*) 'after repoel, np=',np
C    former le nouvel hyperplan de base en tenant compte du point trouve
   IF (PT.GT.0) THEN
      I = I + 1
      INITIA (I) = PT
      POINT1 (PT) = 1
      DO J=1,DIM
         BASE (J,I) = POINT2 (J,PT)
   70       ENDDO
C
C       recherche des coefficients de l'equation de l'hyperplan de base
      CALL REEQHY (BASE,FACE(1,1),HYPERPLAN(1,1),A,B,X,Y,
     +                    DIMMAX,DIM,TERIND,PREC1)
C
c       write(MSG_UNIT,*) 'after REEQHY'
C
   ELSE
      EGAL = .FALSE.
   ENDIF
   GOTO 60
ENDIF
C
C recherche du DIM+1 ieme sommet de base
C
IF (EGAL) THEN
C
C    rechercher le point le plus eloigne de l'hyperplan de base
```

```
      PT = REPOEL (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,1),
     +                  TERIND)
c     write(MSG_UNIT,*) 'after repoel, np=',np
      IF (PT.GT.0) THEN
         I = I + 1
         INITIA (I) = PT
         POINT1 (PT) = 1
      ENDIF
   ENDIF
C
C s'il n'est pas possible de construire le polyedre de base
C
c write (MSG_UNIT,*) ' #pts polyedre base:',i
   IF (I.LT.DIM+1) THEN
      VOLUME = 0.
      IF (I.EQ.DIM) THEN
         NH = 1
         NF = 1
         J = 0
         DO I=1,NP
            IF (POINT1(I).EQ.-1) THEN
               POINT1(I) = 0
            ELSE
               J = J + 1
               FACE(J,1) = I
            ENDIF
      80       ENDDO
         FACE1(I) = 0
         HYPER1(1,1) = TERIND
         HYPER1(2,1) = 1
         HYPER1(3,1) = 1
      ELSE
         NH = 0
         NF = 0
         DO I=1,NP
            POINT1(I) = 0
      85       ENDDO
      ENDIF
   ELSE
C
      NPS = DIM + 1


c
C     ------------------------------------------------
```

```
C    tri par ordre croissant des DIM+1 sommets de base
C    ------------------------------------------------
C
   DO I=DIM,1,-1
      DO J=1,I
         IF (INITIA(J).GT.INITIA(J+1)) THEN
            K = INITIA (J)
            INITIA (J) = INITIA (J+1)
            INITIA (J+1) = K
         ENDIF
  100       ENDDO
   90    ENDDO
C
C    ------------------------------------------------------------
C    determination du centre de gravite et translation des points
C    ------------------------------------------------------------
C
C    determination du centre de gravite
C
   DO I=1,DIM+1
      K = INITIA (I)
      DO 120 J=1,DIM
         VECTRA (J) = VECTRA (J) + POINT2 (J,K)
  120       ENDDO
  110    ENDDO
C
   DO I=1,DIM
      VECTRA (I) = VECTRA (I) / (DIM+1)
      cg (i) = vectra (i)
  130    ENDDO
C
C    translation des points
C
   DO 140 I=1,NP
      DO 150 J=1,DIM
         POINT2 (J,I) = POINT2 (J,I) - VECTRA (J)
  150       CONTINUE
  140    CONTINUE
C
c    write(MSG_UNIT,*) 'VECTRA =',(vectra(j),j=1,dim)
C
C    ------------------------------------
C    calcul du volume du polyedre de base
C    ------------------------------------
C
```

```
      VOLUME = 0.
      K = INITIA (DIM+1)
      DO 160 I=1,DIM
         L = INITIA (I)
         DO 170 J=1,DIM
            A (J,I) = POINT2 (J,L) - POINT2 (J,K)
  170       CONTINUE
  160    CONTINUE
      CALL DGEFA (A,DIM,X,I)
C
c     write(MSG_UNIT,*) 'after DGEFA'
C
      IF (I.EQ.0) THEN
         CALL DGEDI (A,DIM,X,DET,B,10)
         VOLUME = VOLUME + ABS(DET(1)*10.0**DET(2))
      ENDIF


c           write(*,*)'volume=',volume
C     -------------------------------------
C     determination des DIM+1 faces de base
C     -------------------------------------
C
      NF = DIM + 1
      DO 180 I=1,NF
         FACE1 (I) = 0
         DO 190 J=1,DIM
            IF (J.LT.I) THEN
               FACE (J,I) = INITIA (J)
            ELSE
               FACE (J,I) = INITIA (J+1)
            ENDIF
  190       CONTINUE
  180    CONTINUE

c
C     ----------------------------------------
C     determination des DIM+1 hyperplans de base
C     ----------------------------------------
C
      NH  = NF
      NHT = NH
      NHI = 0
      NHF = 0
      DO 200 I=1,NH
```

```
      HYPER1 (1,I) = 1
      HYPER1 (2,I) = 1
      HYPER1 (3,I) = I
      CALL REEQHY (POINT2,FACE(1,I),HYPERPLAN(1,I),A,B,X,Y,
     +                    DIMMAX,DIM,TERIND,PREC1)
C
C       write(MSG_UNIT,*) 'after REEQHY'
C
  200    CONTINUE
C
C    ----------------------------
C    recherche des points internes
C    ----------------------------
C
   DO 210 I=1,NH
      LIHYIN (I) = I
  210    CONTINUE
C
   NPI = 0
   DO 220 I=1,NP
      IF (POINT1(I).EQ.-1) THEN
         IF (INT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,NH,
     +                   LIHYIN,PREC)) THEN
            POINT1 (I) = 0
            NPI = NPI + 1
         ENDIF
      ENDIF
  220    CONTINUE
```

```
C
C ***************************************************************************
C *  CONSTRUCTION DE L'ENVELOPPE CONVEXE :                                  *
C *                                                                         *
C *     - tant qu'il y a des hyperplans transitoires et des points          *
C *       a classer faire:      *
C *        - recherche du premier hyperplan transitoire : HT                *
C *        - recherche du point exterieur a l'hyperplan HT : PT             *
C *        - s'il n'existe pas de point exterieur :                         *
C *           - considerer l'hyperplan HT comme etant final                 *
```

```
C *        - s'il existe un point exterieur :                          *
C *           - considerer le point PT comme etant un sommet           *
C *           - rechercher les hyperplans pour lesquels PT est exte-   *
C *             rieur, les considerer comme etant internes. Rechercher *
C *             les faces pour lesquelles PT est exterieur             *
C *           - recherche des nouvelles faces                          *
C *           - ajouter les nouvelles faces aux faces existantes ainsi *
C *             que leur hyperplan (s'il n'existe pas encore) ou en le *
C *             mettant a jour (s'il existe deja)                      *
C *           - recherche des nouveaux points internes                 *
C ***********************************************************************
C
   HT = 1
   IF ( DIM .GE. 7) THEN
IF (NP .LE. 10) THEN
NP_LIMIT_RATIO = NP
ELSE
NP_LIMIT_RATIO = ( NP * LIMIT_RATIO ) / 100
ENDIF
   ELSE
NP_LIMIT_RATIO = NP
   ENDIF
c    write (MSG_UNIT,'(a,7i5)') ' CONVEXE> npi, nps, np, nhi,
c     +          nhf, nht, nh',npi,nps,np,nhi,nhf,nht,nh
   DO WHILE ((NHT.NE.0).AND.(NPS+NPI.LT.NP_LIMIT_RATIO))
C
C       ---------------------------------
C       purger si besoin est les tableaux
C       ---------------------------------
C
      IF (NHI.GE.500) THEN
c  WRITE (MSG_UNIT,*) ' CONVEXE> calling purge (NHI=500)
c     +                with param:'
C  WRITE (MSG_UNIT,*) '         NH,NHT,NHF,NF:',
C      +                              NH, NHT, NHF, NF
C
         CALL PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,
     +                        HYPER1,FACE1,DIM,DIMMAX)
C
CMHP  Purge problem
CMHP  -------------------
 IF (NH .EQ. 0) THEN
    VOLUME = 0
    RETURN
 ENDIF
```

```
CMHP  ------------------
         HT = 1
         NHI = 0
      ENDIF
C
C       ----------------------------------------
C       recherche du premier hyperplan transitoire
C       ----------------------------------------
C

      DO WHILE (HYPER1(1,HT).NE.1)
         HT = HT + 1
      ENDDO

C       ----------------------------------------------
C       recherche du point exterieur a l'hyperplan HT
C       ----------------------------------------------
C
      PT = REPOEX (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,HT))
C
c       write(MSG_UNIT,*) 'after REPOEX'
C
C       ----------------------------------
C       s'il n'existe pas de point exterieur
C       ----------------------------------
C
      IF (PT.EQ.0) THEN
C
C          ----------------------------------------
C          considerer l'hyperplan HT comme etant final
C          ----------------------------------------
C
         HYPER1 (1,HT) = 2
         NHT = NHT - 1
         NHF = NHF + 1
C

c                write(*,*)'hyper1(1,',ht,')=',hyper1(1,ht)
C       ----------------------------
C       s'il existe un point exterieur
C       ----------------------------
C
      ELSE
C
C          ----------------------------------------
```

```
C          considerer le point PT comme etant un sommet
C          ----------------------------------------
C
c                write(*,*)'pt=',pt

         POINT1 (PT) = 1
         NPS = NPS + 1

c                write(*,*)'nps=',nps
C
C          ------------------------------------------------------------
C          recherche des hyperplans pour lesquels PT est exterieur, les
C             considerer comme etant internes
C          recherche des faces ...
C          ------------------------------------------------------------
C
         NFE = 0
         NH1 = 0
         NH2 = 0
C
C          pour chaque hyperplan
C
         DO I=1,NH
C
C             si l'hyperplan I est transitoire
C
            IF (HYPER1(1,I).EQ.1) THEN
               SOMME = SOM (DIM,POINT2(1,PT),HYPERPLAN(1,I))
C
C                 si le point PT est exterieur a l'hyperplan I
C
CMH                 IF (SOMME.GT.1.) THEN
               IF (SOMME.GT.(1.+PREC)) THEN
C
C                    considerer l'hyperplan I comme etant interne
C
                  HYPER1 (1,I) = 0
                  NHT = NHT - 1
                  NHI = NHI + 1
C
C                    ajouter l'hyperplan I aux hyperplans exterieurs
C                    pour PT
C
                  NH2 = NH2 + 1
  IF ( NH2 .GE. FATAL_LIMIT_NHEMAX) THEN
```

```
WRITE (MSG_UNIT,*) ' CONVEXE> NHEMAX!...'
STOP
  ENDIF
                  LIHYEX (NH2) = I
C                   ajouter les faces aux faces exterieures pour PT
C
                  J = HYPER1 (3,I)
                  DO WHILE (J.NE.0)
                     NFE = NFE + 1
     IF ( NFE .GE. FATAL_LIMIT_NFEMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NFEMAX!...'
STOP
     ENDIF
                     FACEXT (NFE) = J
                     J = FACE1 (J)
                  ENDDO
               ENDIF
            ENDIF
            ENDDO
C
C           ------------------------------------------
C           calcul du volume ajoute au polyedre courant
C           ------------------------------------------
C
         DO 270 I=1,NFE
            L = FACEXT (I)
            DO 275 k=1,DIM
               cgaux (k) = point (k,pt)
  275          CONTINUE
            DO 280 J=1,DIM
               M = FACE (J,L)
               DO 290 K=1,DIM
                  A (K,J) = POINT2 (K,M) - POINT2 (K,PT)
                  cgaux (k) = cgaux (k) + point (k,m)
  290             CONTINUE
  280          CONTINUE
            DO 285 J=1,DIM
               cgaux (j) = cgaux (j)/(dim+1)
  285          CONTINUE
            CALL DGEFA (A,DIM,X,J)
            IF (J.EQ.0) THEN
               CALL DGEDI (A,DIM,X,DET,B,10)
C
c        write(MSG_UNIT,*) 'after DGEDI'
C
```

```
               vol = ABS(DET(1)*10.0**DET(2))
               DO 287 k=1,DIM
                  cg(k)=((cg(k)*volume)+(cgaux(k)*vol))/
     +         (volume+vol)
  287               CONTINUE
               VOLUME = VOLUME + vol
            ENDIF
  270          CONTINUE
C
C           ----------------------------
C           recherche des nouvelles faces
C           ----------------------------
C
         CALL RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,
     +                       DIM,NNF,NOUFAC)
C
C  write(MSG_UNIT,*) 'after renofa'
C
C           ------------------------------------------------------------
C           ajouter les nouvelles faces aux faces existantes ainsi que
C           leur hyperplan (s'il n'existe pas encore) ou en le mettant a
C           jour (s'il existe deja)
C           ------------------------------------------------------------
 FATAL_CURRENT_NF = FATAL_CURRENT_NF + NNF

c                write(*,*)'FATAL_CURRENT_NF=',FATAL_CURRENT_NF
c                write(*,*)'nnf=',nnf
c                WRITE(*,*)'FATAL_LIMIT_NFMAX=',FATAL_LIMIT_NFMAX

 IF ( FATAL_CURRENT_NF .GE. FATAL_LIMIT_NFMAX) THEN
    WRITE (MSG_UNIT,*) ' CONVEXE> NFMAX!...'
    STOP
 ENDIF
         DO 300 I=1,NNF
            NF = NF + 1
            DO 310 J=1,DIM
               FACE (J,NF) = NOUFAC (J,I)
  310             CONTINUE
C
C             calcul de l'equation de l'hyperplan contenant la nouvelle
C             face
C
            CALL REEQHY (POINT2,FACE(1,NF),HYPERPLAN(1,NH+1),
     +                           A,B,X,Y,DIMMAX,DIM,TERIND,PREC1)
C
```

```
C
C             voir si l'hyperplan existe deja ou non
C
            EGAL = .FALSE.
            J = NH
  320             CONTINUE
            IF ((.NOT.EGAL).AND.(J.GT.0)) THEN
               K = 0
  330                CONTINUE
                  K = K + 1
                  EGAL = ABS(HYPERPLAN(K,J)-HYPERPLAN(K,NH+1))
     +                           .LE.PREC2
               IF (EGAL.AND.(K.LT.DIM)) GOTO 330
               J = J - 1
               GOTO 320
            ENDIF
C
C             il y aura un hyperplan de plus, interieur pour PT
C
            NH1 = NH1 + 1
IF ( NH1 .GE. FATAL_LIMIT_NHEMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NHEMAX!...'
STOP
ENDIF
C
C             si l'hyperplan existe deja
C
            IF (EGAL) THEN
               J = J + 1
               FACE1 (NF) = HYPER1 (3,J)
               HYPER1 (2,J) = HYPER1 (2,J) + 1
               HYPER1 (3,J) = NF
               LIHYIN (NH1) = J
C
C             si l'hyperplan n'existe pas encore
C
            ELSE
               NH = NH + 1
       FATAL_CURRENT_NH = FATAL_CURRENT_NH + 1
       IF ( FATAL_CURRENT_NH .GE. FATAL_LIMIT_NHMAX) THEN
   WRITE (MSG_UNIT,*) ' CONVEXE> NHMAX!...'
   STOP
       ENDIF
               NHT = NHT + 1
               FACE1 (NF) = 0
```

```
               HYPER1 (1,NH) = 1
               HYPER1 (2,NH) = 1
               HYPER1 (3,NH) = NF
               LIHYIN (NH1) = NH
            ENDIF
  300          CONTINUE
C
C           ------------------------------------
C           recherche des nouveaux points internes
C           ------------------------------------
C
C
c  write(MSG_UNIT,*) 'after recherche points internes'
C
         DO I=1,NP
            IF (POINT1(I).EQ.-1) THEN
               IF (EXT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,NH2,
     +                          LIHYEX)) THEN
                  IF (INT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,
     +                             NH1,LIHYIN,PREC)) THEN
                     POINT1 (I) = 0
                     NPI = NPI + 1
                  ENDIF
               ENDIF
            ENDIF
            ENDDO
      ENDIF
   ENDDO
C
C ***********************************************************************
C *  CLOTURE :                                                          *
C *                                                                     *
C *     - considerer les hyperplans transitoires comme finaux           *
C *     - purge des hyperplans, des faces et nettoyage des tableaux     *
C *     - translation des points                                        *
C *     - adaptation des equations des hyperplans                       *
C ***********************************************************************
C
C    ------------------------------------------------
C    considerer les hyperplans transitoires comme finaux
C    ------------------------------------------------
C
   DO  I=1,NH
      IF (HYPER1(1,I).EQ.1) THEN
         HYPER1 (1,I) = 2
```

```
      ENDIF
      ENDDO
   NHF = NHF + NHT
       NHT = 0
C
C    ------------------------------------------------------
C    calcul de la surface du convexe
C    ------------------------------------------------------
C
   SURFACE=0
   DO I=1,NH
     IF (HYPER1(1,I).EQ.2) THEN
       SOMME=0
       DO K=1,DIM
         SOMME = SOMME + HYPERPLAN(K,I)*HYPERPLAN(K,I)
       ENDDO
       SOMME = SQRT(SOMME)
       DO K=2,DIM-1
         SOMME=SOMME/K
       ENDDO
       J=HYPER1(3,I)
       DO WHILE (J.NE.0)
C
C          calcul de la surface de la face J
C
         DO K=1,DIM
           DO L=1,DIM
             A(K,L) = POINT2(L,FACE(K,J))
           ENDDO
 ENDDO

C          --- calcul du determinant de A

         AUX=0
           CALL DGEFA (A,DIM,X,K)
C
c  write(MSG_UNIT,*) 'after DGEFA'
C
         IF (K.EQ.0) THEN
           CALL DGEDI(A,DIM,X,DET,B,10)
           AUX=ABS(DET(1)*10.0**DET(2))
         ELSE
           WRITE(MSG_UNIT,*) 'CONVEXE> WARNING : face de surface
     +                          nulle'
         ENDIF
```

```
          SURFACE = SURFACE + AUX*SOMME
          J=FACE1(J)
        ENDDO
      ENDIF
    ENDDO
c           write(*,*)'surface=', surface



C     ------------------------------------------------------
C     purge des hyperplans, des faces et nettoyage des tableaux
C     ------------------------------------------------------
C
c     WRITE (MSG_UNIT,*) ' CONVEXE> calling purge (end) with param:'
c     WRITE (MSG_UNIT,*) '         NH,NHT,NHF,NF:',NH,NHT,NHF,NF
    CALL PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,HYPER1,
     +          FACE1,DIM,DIMMAX)
C
C     write(MSG_UNIT,*) 'after purge'
C
CMHP     Purge problem
CMHP     -------------------
    IF (NH .EQ. 0) THEN
VOLUME = 0
RETURN
    ENDIF
CMHP     -------------------
C
C     --------------------------------------
C     adaptation des equations des hyperplans
C     --------------------------------------
C
    IF (.NOT.COND) THEN
       DO I=1,NH
          AUX = 1.
          DO J=1,DIM
             AUX = AUX + HYPERPLAN (J,I) * VECTRA (J)
             ENDDO
          IF (ABS(AUX).GT.1E-6) THEN
             DO J=1,DIM
                HYPERPLAN (J,I) = HYPERPLAN (J,I) / AUX
                ENDDO
             HYPER1 (1,I) = 1
          ELSE
```

```
            HYPER1 (1,I) = 0
         ENDIF
         ENDDO
      DO I=1,DIM
         VECTRA (I) = 0.
         ENDDO
   ELSE
      DO I=1,NH
         AUX = 1.
         DO J=1,DIM
            AUX = AUX + HYPERPLAN (J,I) * (VECTRA(J)-cg(j))
            ENDDO
         IF (ABS(AUX).GT.1E-6) THEN
            DO J=1,DIM
               HYPERPLAN (J,I) = HYPERPLAN (J,I) / AUX
               ENDDO
            HYPER1 (1,I) = 1
         ELSE
            HYPER1 (1,I) = 0
         ENDIF
         ENDDO
      DO I=1,DIM
         VECTRA (I) = cg (i)
         ENDDO
   ENDIF
C
C    --------------------
C    ajustement du volume
C    --------------------
C
   AUX = 1.
   DO I=2,DIM
     AUX=AUX*I
   ENDDO
   VOLUME = VOLUME / AUX
C
ENDIF
C


c        WRITE (MSG_UNIT,*) ' CONVEXE> # hyperplans, volume', NH, VOLUME
c write (MSG_UNIT,*) 'CG=',(cg(j),j=1,dim)
c
END
C
```

```
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C
DOUBLE PRECISION FUNCTION SOM (DIM,POINT,HYPERPLAN)
C
C       SOM (DIM,POINT,HYPERPLAN) : fonction DOUBLE PRECISION qui
C     calcule le produit scalaire du vecteur POINT avec le
C     vecteur HYPERPLAN, les deux vecteurs etant de dimension DIM.
C
C
IMPLICIT NONE
C
INTEGER          DIM
DOUBLE PRECISION POINT(*),HYPERPLAN(*)
C
INTEGER          I
DOUBLE PRECISION SOMME
C
SOMME = 0.
DO 10 I=1,DIM
   SOMME = SOMME + POINT(I) * HYPERPLAN(I)
   10 CONTINUE
SOM = SOMME
C
END
C
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C -----------------------------------------------------------------------
C
INTEGER FUNCTION REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,
     +    TERIND)
C
C
C       REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,TERIND) : fonction
C           entiere qui renvoie l'indice du point le plus eloigne de
C           l'hyperplan specifie. La recherche du point ne s'effectue que
C           sur les points exterieurs au polyedre courant.
C
IMPLICIT NONE
C
INTEGER          NP,DIM,DIMMAX,POINT1(*),TERIND
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*)
C
```

```
INTEGER          I,PT
DOUBLE PRECISION SOMME,DISTANCE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
INTRINSIC ABS
C
PT = 0
DISTANCE = 0.
DO 10 I=1,NP
   IF (POINT1(I).EQ.-1) THEN
      SOMME = SOM (DIM,POINT(1,I),HYPERPLAN)
      IF (ABS(SOMME-TERIND).GT.(DISTANCE+1.D-10)) THEN
         PT = I
         DISTANCE = ABS(SOMME-TERIND)
      ENDIF
   ENDIF
   10 CONTINUE
REPOEL = PT
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
INTEGER FUNCTION REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN)
C
C
C REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN) : fonction entiere
C     qui renvoie l'indice du point le plus exterieur
C     a l'hyperplan specifie. La recherche du point ne s'
C     effectue que sur les points exterieurs au polyedre courant.
C
IMPLICIT NONE
C
INTEGER          NP,DIM,DIMMAX,POINT1(*)
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*)
C
INTEGER          I,PT
DOUBLE PRECISION SOMME,DISTANCE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
```

```
PT = 0
DISTANCE = 0.
DO 10 I=1,NP
   IF (POINT1(I).EQ.-1) THEN
      SOMME = SOM (DIM,POINT(1,I),HYPERPLAN)
      IF ((SOMME-1.).GT.DISTANCE) THEN
         PT = I
         DISTANCE = SOMME - 1.
      ENDIF
   ENDIF
   10 CONTINUE
REPOEX = PT
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
LOGICAL FUNCTION INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP,
     +    PREC)
C
C
C INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est interieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C
C
IMPLICIT NONE
C
INTEGER          DIMMAX,DIM,NH,LISHYP(*)
DOUBLE PRECISION POINT(*),HYPERPLAN(DIMMAX,*),PREC
C
INTEGER          I
DOUBLE PRECISION SOMME
LOGICAL          VALABLE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
VALABLE = .TRUE.
C
I = 1
   10 CONTINUE
IF (VALABLE.AND.(I.LE.NH)) THEN
```

```
   SOMME = SOM (DIM,POINT,HYPERPLAN(1,LISHYP(I)))
   VALABLE = SOMME.LE.(1.+PREC)
   I = I + 1
   GOTO 10
ENDIF
C
INT = VALABLE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
LOGICAL FUNCTION EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP)
C
C
C EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est exterieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C
C
IMPLICIT NONE
C
INTEGER          DIMMAX,DIM,NH,LISHYP(*)
DOUBLE PRECISION POINT(*),HYPERPLAN(DIMMAX,*)
C
INTEGER          I
DOUBLE PRECISION SOMME
LOGICAL          VALABLE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
VALABLE = .FALSE.
C
I = 1
   10 CONTINUE
IF ((.NOT.VALABLE).AND.(I.LE.NH)) THEN
   SOMME = SOM (DIM,POINT,HYPERPLAN(1,LISHYP(I)))
   VALABLE = SOMME.GT.1.
   I = I + 1
   GOTO 10
ENDIF
C
```

```
EXT = VALABLE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE RENOFA (MSG_UNIT,PT,FACEXT,NFE,FACE,DIMMAX,
     +                     DIM,NNF,NOUFAC)
C
C
C RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,DIM,NNF,NOUFAC) :
C     sous-routine qui recherche les nouvelles faces engendrees par le
C           point PT et par les NFE faces dont les indices se trouvent dans
C           le tableau FACEXT. Cette routine exclut les faces internes.
C
C
IMPLICIT NONE

INCLUDE 'FATAL_LIMITS.DEF'
C
INTEGER PT,DIMMAX,FACEXT(*),NFE,FACE(DIMMAX,*),DIM,NNF
INTEGER NOUFAC(DIMMAX,*),MSG_UNIT
C
INTEGER I,J,K,L,M,IND
LOGICAL EXISTE
C
NNF = 0
DO 10 I=1,NFE
   IND = FACEXT(I)
   DO 20 J=1,DIM
      DO 30 K=1,DIM
         NOUFAC (K,NNF+1) = FACE (K,IND)
30       CONTINUE
      NOUFAC (J,NNF+1) = PT
C
      DO 40 K=DIM-1,1,-1
         DO 50 L=1,K
            IF (NOUFAC(L,NNF+1).GT.NOUFAC(L+1,NNF+1)) THEN
               M = NOUFAC(L,NNF+1)
               NOUFAC(L,NNF+1) = NOUFAC(L+1,NNF+1)
               NOUFAC(L+1,NNF+1) = M
            ENDIF
50          CONTINUE
```

```
   40       CONTINUE
C
C       verifier si nouvelle face existe deja ou non. Si oui, il s'agit
C       d'une face interne, il faut oublier cette face ainsi que sa
C       deuxieme occurence.
C
        EXISTE = .FALSE.
        K = 0
   60       CONTINUE
        IF ((.NOT.EXISTE).AND.(K.LT.NNF)) THEN
           K = K + 1
           L = 0
   70          CONTINUE
              L = L + 1
              EXISTE = NOUFAC(L,NNF+1).EQ.NOUFAC(L,K)
           IF (EXISTE.AND.(L.LT.DIM)) GOTO 70
           GOTO 60
        ENDIF
        IF (EXISTE) THEN
           DO 80 L=K+1,NNF
              DO 90 M=1,DIM
                 NOUFAC (M,L-1) = NOUFAC (M,L)
   90               CONTINUE
   80          CONTINUE
           NNF = NNF - 1
        ELSE
           NNF = NNF + 1
 IF ( NNF .GE. FATAL_LIMIT_NNFMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NNFMAX!...'
STOP
 ENDIF
        ENDIF
C
   20    CONTINUE
   10 CONTINUE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,
     +    DIM,TERIND,PREC)
C
```

```
C
C REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,DIM,TERIND,PREC) :
C     sous-routine qui recherche l'equation d'un hyperplan.
C     Plus particulierement, elle recherche les coefficients de
C     l'hyperplan devant contenir les points de la face specifiee
C     Les coefficients obtenus constituent l'hyperplan
C     specifie. La variable TERIND contient la valeur du terme
C     independant de l'equation de l'hyperplan.
C
C
IMPLICIT NONE
C
INTEGER          FACE(*),X(*),Y(*),DIMMAX,DIM,TERIND
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*),A(DIMMAX,*)
        DOUBLE PRECISION B(*),PREC
C
INTEGER          I,J,K,L,PT,NCSP
DOUBLE PRECISION AUX
C
INTRINSIC ABS
C
C initialisation des vecteurs A, B, X et Y
C
DO 10 I=1,DIM
   B (I) = 1.
   PT = FACE (I)
   DO 20 J=1,DIM
      A (J,I) = POINT (J,PT)
   20    CONTINUE
   X (I) = 0
   Y (I) = 0
   10 CONTINUE
TERIND = 1
C
NCSP = 0
DO 30 J=1,DIM
C
C    recherche du pivot de la Jieme colonne
C
   I = 0
   AUX = 0.
   DO 40 K=1,DIM
      IF (X(K).EQ.0) THEN
         IF ((A(J,K).NE.0.).AND.(ABS(A(J,K)).GT.AUX)) THEN
            AUX = ABS(A(J,K))
```

```
            I = K
         ENDIF
      ENDIF
   40    CONTINUE
C
   IF (I.NE.0) THEN
      X (I) = J
C
C       diviser la ligne du pivot par celui-ci
C
      AUX = A (J,I)
      DO 50 K=1,DIM
         A (K,I) = A (K,I) / AUX
   50       CONTINUE
      B (I) = B (I) / AUX
C
C       pivotage
C
      DO 60 K=1,I-1
         AUX = A (J,K)
         DO 70 L=1,DIM
            A (L,K) = A (L,K) - AUX * A (L,I)
   70          CONTINUE
         B (K) = B (K) - AUX * B (I)
   60       CONTINUE
      DO 80 K=I+1,DIM
         AUX = A (J,K)
         DO 90 L=1,DIM
            A (L,K) = A (L,K) - AUX * A (L,I)
   90          CONTINUE
         B (K) = B (K) - AUX * B (I)
   80       CONTINUE
   ELSE
      NCSP = NCSP + 1
      Y (NCSP) = J
   ENDIF
   30 CONTINUE
C
C voir si on a une matrice singuliere ou non
C
K = 1
IF (NCSP.NE.0) THEN
   TERIND = 0
   DO 100 I=1,DIM
      B (I) = 0.
```

```
      IF (X(I).NE.0) THEN
         DO 110 J=1,NCSP
            B (I) = B (I) - A (Y(J),I)
  110          CONTINUE
      ELSE
         B (I) = 1.
         X (I) = Y (K)
         K = K + 1
      ENDIF
  100    CONTINUE
ENDIF
C
C reorganisation du vecteur B et des coefficients solutions
C
DO 120 I=1,DIM
   A (X(I),1) = B (I)
  120 CONTINUE
DO 130 I=1,DIM
   IF (ABS(A(I,1)).LT.PREC) THEN
      HYPERPLAN (I) = 0.
   ELSE
      HYPERPLAN (I) = A (I,1)
   ENDIF
  130 CONTINUE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,
     +                  HYPER1,FACE1,DIM,DIMMAX)
C
C       sous-routine qui purge les hyperplans et les faces, i.e. stocke les
C       hyperplans transitoires et finaux, ainsi que les faces qui leur
C       correspondent; et nettoyage des tableaux.
C
C
IMPLICIT NONE
C

INTEGER NH,NHT,NHF,NF,DIM,DIMMAX,FACE(DIMMAX,*),HYPER1(3,*)
INTEGER FACE1(*),MSG_UNIT
DOUBLE PRECISION HYPERPLAN(DIMMAX,*)
```

```
C
INTEGER I,J,K,L,M
C
C purge des hyperplans
C
K = 0
DO 10 I=1,NH
   IF ((HYPER1(1,I).EQ.1).OR.(HYPER1(1,I).EQ.2)) THEN
      K = K + 1
      DO 20 J=1,DIM
         HYPERPLAN (J,K) = HYPERPLAN (J,I)
   20       CONTINUE
      DO 30 J=1,3
         L = HYPER1 (J,I)
         HYPER1 (J,I) = HYPER1 (J,K)
         HYPER1 (J,K) = L
   30       CONTINUE
   ENDIF
   10 CONTINUE
C
c WRITE (MSG_UNIT,*) ' PURGE> # hyp finaux:', K
C
C
C purge des faces
C
DO 40 I=NHF+NHT+1,NH
   J = HYPER1 (3,I)
   50    CONTINUE
      K = FACE1 (J)
      FACE1 (J) = -1
      J = K
   IF (J.NE.0) GOTO 50
   40 CONTINUE
C
K = 1
DO 60 I=1,NHF+NHT
c           write(*,*)'i=',i
   J = HYPER1 (3,I)
c           write(*,*)'j=',j

   HYPER1 (3,I) = K
   70    CONTINUE
      IF (K.NE.J) THEN
         IF (FACE1(K).EQ.-1) THEN
            DO 80 L=1,DIM
```

```
               FACE (L,K) = FACE (L,J)
  80              CONTINUE
               FACE1 (K) = FACE1 (J)
               FACE1 (J) = -1
            ELSE
               L = I + 1
  90              CONTINUE
               IF (L.LT.NHF+NHT) THEN
                  IF (HYPER1(3,L).NE.K) THEN
                     L = L + 1
                  ELSE
                     HYPER1 (3,L) = J
                     L = NH + 1
                  ENDIF
                  GOTO 90
               ENDIF
               IF (L.EQ.NHF+NHT) THEN
                  L = K + 1
  100                CONTINUE
                  IF (L.LT.NF) THEN
                     IF (FACE1(L).NE.K) THEN
                        L = L + 1
                     ELSE
                        FACE1 (L) = J
                        L = NF + 1
                     ENDIF
                     GOTO 100
                  ENDIF
               ENDIF
c                    write(*,*)'j et k=',j,k

CMHP     Purge problem
CMHP     ------------------------------------------------
C     WRITE (MSG_UNIT,*) ' PURGE> permute faces - J,K:', J, K
      IF (J .LT. K) THEN
WRITE (MSG_UNIT,*) ' PURGE> # Hyper forced to 0'
NH  = 0
NHF = 0
NHT = 0
NF  = 0
RETURN
      ENDIF
CMHP     ------------------------------------------------
               DO 110 L=1,DIM
                  M = FACE (L,J)
```

```
                FACE (L,J) = FACE (L,K)
                FACE (L,K) = M
  110              CONTINUE
              L = FACE1 (J)
              FACE1 (J) = FACE1 (K)
              FACE1 (K) = L
           ENDIF
        ENDIF
        J = FACE1 (K)
        K = K + 1
     IF (J.NE.0) THEN
        FACE1 (K-1) = K
        GOTO 70
     ENDIF
     60 CONTINUE
C
C nettoyage des tableaux
C
DO 120 I=NHF+NHT+1,NH
   DO 130 J=1,DIMMAX
      HYPERPLAN (J,I) = 0.
  130    CONTINUE
   DO 140 J=1,3
      HYPER1 (J,I) = 0
  140    CONTINUE
  120 CONTINUE
NH = NHF+NHT
C
DO 150 I=K,NF
   DO 160 J=1,DIMMAX
      FACE (J,I) = 0
  160    CONTINUE
   FACE1 (I) = 0
  150 CONTINUE
NF = K - 1
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C

      SUBROUTINE DGEDI(A,N,IPVT,DET,WORK,JOB)
C
```

```
      INCLUDE 'CONSTMAX.DEF'
C
      INTEGER N,IPVT(1),JOB
      DOUBLE PRECISION A(DIMMAX,1),DET(2),WORK(1)
C
C     DGEDI COMPUTES THE DETERMINANT AND INVERSE OF A MATRIX
C     USING THE FACTORS COMPUTED BY DGECO OR DGEFA.
C
C     ON ENTRY
C
C        A       DOUBLE PRECISION(DIMAX,N)
C                THE OUTPUT FROM DGECO OR DGEFA.
C
C        DIMMAX  INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C        IPVT    INTEGER(N)
C                THE PIVOT VECTOR FROM DGECO OR DGEFA.
C
C        WORK    DOUBLE PRECISION(N)
C                WORK VECTOR.  CONTENTS DESTROYED.
C
C        JOB     INTEGER
C                = 11   BOTH DETERMINANT AND INVERSE.
C                = 01   INVERSE ONLY.
C                = 10   DETERMINANT ONLY.
C
C     ON RETURN
C
C        A       INVERSE OF ORIGINAL MATRIX IF REQUESTED.
C                OTHERWISE UNCHANGED.
C
C        DET     DOUBLE PRECISION(2)
C                DETERMINANT OF ORIGINAL MATRIX IF REQUESTED.
C                OTHERWISE NOT REFERENCED.
C                DETERMINANT = DET(1) * 10.0**DET(2)
C                WITH  1.0 .LE. DABS(DET(1)) .LT. 10.0
C                OR  DET(1) .EQ. 0.0 .
C
C     ERROR CONDITION
C
C        A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS
```

```
C        A ZERO ON THE DIAGONAL AND THE INVERSE IS REQUESTED.
C        IT WILL NOT OCCUR IF THE SUBROUTINES ARE CALLED CORRECTLY
C        AND IF DGECO HAS SET RCOND .GT. 0.0 OR DGEFA HAS SET
C        INFO .EQ. 0 .
C
C     LINPACK. THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS DAXPY,DSCAL,DSWAP
C     FORTRAN DABS,MOD
C
C     INTERNAL VARIABLES
C
      DOUBLE PRECISION T
      DOUBLE PRECISION TEN
      INTEGER I,J,K,KB,KP1,L,NM1
C
C
C     COMPUTE DETERMINANT
C
      IF (JOB/10 .EQ. 0) GO TO 70
         DET(1) = 1.0D0
         DET(2) = 0.0D0
         TEN = 10.0D0
         DO 50 I = 1, N
            IF (IPVT(I) .NE. I) DET(1) = -DET(1)
            DET(1) = A(I,I)*DET(1)
C           ...EXIT
            IF (DET(1) .EQ. 0.0D0) GO TO 60
   10       IF (DABS(DET(1)) .GE. 1.0D0) GO TO 20
               DET(1) = TEN*DET(1)
               DET(2) = DET(2) - 1.0D0
            GO TO 10
   20       CONTINUE
   30       IF (DABS(DET(1)) .LT. TEN) GO TO 40
               DET(1) = DET(1)/TEN
               DET(2) = DET(2) + 1.0D0
            GO TO 30
   40       CONTINUE
   50    CONTINUE
   60    CONTINUE
   70 CONTINUE
C
```

```
C     COMPUTE INVERSE(U)
C
      IF (MOD(JOB,10) .EQ. 0) GO TO 150
         DO 100 K = 1, N
            A(K,K) = 1.0D0/A(K,K)
            T = -A(K,K)
            CALL DSCAL(K-1,T,A(1,K),1)
            KP1 = K + 1
            IF (N .LT. KP1) GO TO 90
            DO 80 J = KP1, N
               T = A(K,J)
               A(K,J) = 0.0D0
               CALL DAXPY(K,T,A(1,K),1,A(1,J),1)
   80       CONTINUE
   90       CONTINUE
  100    CONTINUE
C
C        FORM INVERSE(U)*INVERSE(L)
C
         NM1 = N - 1
         IF (NM1 .LT. 1) GO TO 140
         DO 130 KB = 1, NM1
            K = N - KB
            KP1 = K + 1
            DO 110 I = KP1, N
               WORK(I) = A(I,K)
               A(I,K) = 0.0D0
  110       CONTINUE
            DO 120 J = KP1, N
               T = WORK(J)
               CALL DAXPY(N,T,A(1,J),1,A(1,K),1)
  120       CONTINUE
            L = IPVT(K)
            IF (L .NE. K) CALL DSWAP(N,A(1,K),1,A(1,L),1)
  130    CONTINUE
  140    CONTINUE
  150 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DGEFA(A,N,IPVT,INFO)
C
      INCLUDE 'CONSTMAX.DEF'
C
```

```
      INTEGER N,IPVT(1),INFO
      DOUBLE PRECISION A(DIMMAX,1)
C
C     DGEFA FACTORS A DOUBLE PRECISION MATRIX BY GAUSSIAN ELIMINATION.
C
C     DGEFA IS USUALLY CALLED BY DGECO, BUT IT CAN BE CALLED
C     DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C     (TIME FOR DGECO) = (1 + 9/N)*(TIME FOR DGEFA) .
C
C     ON ENTRY
C
C        A        DOUBLE PRECISION(DIMMAX, N)
C                 THE MATRIX TO BE FACTORED.
C
C        DIMMAX   INTEGER
C                 THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N        INTEGER
C                 THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A        AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                 WHICH WERE USED TO OBTAIN IT.
C                 THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                 L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                 TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT     INTEGER(N)
C                 AN INTEGER VECTOR OF PIVOT INDICES.
C
C        INFO     INTEGER
C                 = 0  NORMAL VALUE.
C                 = K  IF  U(K,K) .EQ. 0.0 .  THIS IS NOT AN ERROR
C                      CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C                      INDICATE THAT DGESL OR DGEDI WILL DIVIDE BY ZERO
C                      IF CALLED.  USE  RCOND  IN DGECO FOR A RELIABLE
C                      INDICATION OF SINGULARITY.
C
C     LINPACK. THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS DAXPY,DSCAL,IDAMAX
```

```
C
C     INTERNAL VARIABLES
C
      DOUBLE PRECISION T
      INTEGER IDAMAX,J,K,KP1,L,NM1
C
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
C
C        FIND L = PIVOT INDEX
C
         L = IDAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (A(L,K) .EQ. 0.0D0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0D0/A(K,K)
            CALL DSCAL(N-K,T,A(K+1,K),1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            DO 30 J = KP1, N
               T = A(L,J)
               IF (L .EQ. K) GO TO 20
                  A(L,J) = A(K,J)
                  A(K,J) = T
   20          CONTINUE
```

```
                CALL DAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30         CONTINUE
           GO TO 50
   40      CONTINUE
              INFO = K
   50      CONTINUE
   60 CONTINUE
   70 CONTINUE
      IPVT(N) = N
      IF (A(N,N) .EQ. 0.0D0) INFO = N
      RETURN
      END


      SUBROUTINE  DSCAL(N,DA,DX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     USES UNROLLED LOOPS FOR INCREMENT EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DA,DX(1)
      INTEGER I,INCX,M,MP1,N,NINCX
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      NINCX = N*INCX
      DO 10 I = 1,NINCX,INCX
        DX(I) = DA*DX(I)
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20 M = MOD(N,5)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DX(I) = DA*DX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
```

```
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        DX(I) = DA*DX(I)
        DX(I + 1) = DA*DX(I + 1)
        DX(I + 2) = DA*DX(I + 2)
        DX(I + 3) = DA*DX(I + 3)
        DX(I + 4) = DA*DX(I + 4)
   50 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DAXPY(N,DA,DX,INCX,DY,INCY)
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DA
      INTEGER I,INCX,INCY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF (DA .EQ. 0.0D0) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DY(IY) = DY(IY) + DA*DX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
```

```
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DY(I) = DY(I) + DA*DX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        DY(I) = DY(I) + DA*DX(I)
        DY(I + 1) = DY(I + 1) + DA*DX(I + 1)
        DY(I + 2) = DY(I + 2) + DA*DX(I + 2)
        DY(I + 3) = DY(I + 3) + DA*DX(I + 3)
   50 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE  DSWAP (N,DX,INCX,DY,INCY)
C
C     INTERCHANGES TWO VECTORS.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DTEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL
C         TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DTEMP = DX(IX)
        DX(IX) = DY(IY)
        DY(IY) = DTEMP
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
```

```
C       CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C       CLEAN-UP LOOP
C
   20 M = MOD(N,3)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DTEMP = DX(I)
        DX(I) = DY(I)
        DY(I) = DTEMP
   30 CONTINUE
      IF( N .LT. 3 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,3
        DTEMP = DX(I)
        DX(I) = DY(I)
        DY(I) = DTEMP
        DTEMP = DX(I + 1)
        DX(I + 1) = DY(I + 1)
        DY(I + 1) = DTEMP
        DTEMP = DX(I + 2)
        DX(I + 2) = DY(I + 2)
        DY(I + 2) = DTEMP
   50 CONTINUE
      RETURN
      END
      INTEGER FUNCTION IDAMAX(N,DX,INCX)
C
C     FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DMAX
      INTEGER I,INCX,IX,N
C
      IDAMAX = 0
      IF( N .LT. 1 ) RETURN
      IDAMAX = 1
      IF(N.EQ.1)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      IX = 1
      DMAX = DABS(DX(1))
```

```
      IX = IX + INCX
      DO 10 I = 2,N
         IF(DABS(DX(IX)).LE.DMAX) GO TO 5
         IDAMAX = I
         DMAX = DABS(DX(IX))
    5    IX = IX + INCX
   10 CONTINUE
      RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
   20 DMAX = DABS(DX(1))
      DO 30 I = 2,N
         IF(DABS(DX(I)).LE.DMAX) GO TO 30
         IDAMAX = I
         DMAX = DABS(DX(I))
   30 CONTINUE
      RETURN
      END


c     ****************************************************************
c      FATAL_LIMITS.DEF

      INTEGER FATAL_LIMIT_NHMAX, FATAL_LIMIT_NFMAX, FATAL_LIMIT_NPMAX
      INTEGER FATAL_LIMIT_NHEMAX, FATAL_LIMIT_NFEMAX, FATAL_LIMIT_NNFMAX
      INTEGER FATAL_CURRENT_NH, FATAL_CURRENT_NF
C
      COMMON/FATAL_LIMITS1/FATAL_LIMIT_NHMAX, FATAL_LIMIT_NFMAX
      COMMON/FATAL_LIMITS2/FATAL_LIMIT_NPMAX, FATAL_LIMIT_NHEMAX
      COMMON/FATAL_LIMITS3/FATAL_LIMIT_NFEMAX, FATAL_LIMIT_NNFMAX
      COMMON/FATAL_LIMITS4/FATAL_CURRENT_NH, FATAL_CURRENT_NF

c     ****************************************************************
c     CONSTMAX.DEF

INTEGER MSG_UNIT
PARAMETER (MSG_UNIT=6)
C
INTEGER DIMMAX,NPMAX,NHMAX,NFMAX,NCBMAX,NCFMAX,NMAX
C
PARAMETER (DIMMAX=17,NPMAX=120000,NHMAX=50000,NFMAX=50000,
     +            NCBMAX=10,NCFMAX=50000,NMAX=50000)
C
C INTEGER NPMX1
```

```
C PARAMETER (NPMX1=20000)
C
INTEGER NHEMAX,NFEMAX,NNFMAX
PARAMETER (NHEMAX=4000,NFEMAX=4000,NNFMAX=5000)
C
C
C DESCRIPTION DES CONSTANTES
C ************************
C
C DIMMAX : Variable entiere contenant la dimension de declaration des
C  tableaux. Cela correspond a la dimension maximale du probleme.
C NPMAX : Nombre maximal de points.
C NHMAX : Nombre maximal d'hyperplans.
C NFMAX : Nombre maximal de faces.
C NCBMAX : Nombre maximal de clusters
C NCFMAX : Nombre maximal de faces
C NMAX : Nombre maximal de lignes des tableaux TABLE et REF;
C        nombre maximal de colonnes du vecteur CLASSEMENT.
C NPMX1 :
C NHEMAX : Nombre maximal d'hyperplans vis-a-vis desquelles un meme
C          point peut etre exterieur.
C NFEMAX : Nombre maximal de faces vis-a-cvis desquelles un meme
C          point peut etre exterieur.
C NNFMAX : Nombre maximal de nouvelles faces pouvant etre engendrees
C          par un meme point.
C
C
INTEGER IO_SEG_NEXT_FREE_UNIT
COMMON / IO_SEG_FREE_UNIT_COM / IO_SEG_NEXT_FREE_UNIT
```

Annexe B

Programme pour les méthodes issues du classement de Milligan et Cooper

```
c*****************************************************************
c*         Calcul des indices correspondant aux methodes Gamma,    *
c*          de Duda-Heart, de Beale, de Calinski-Harabasz,         *
c*                        et du C-index.                           *
c*****************************************************************

c Declarations:
c---------------
      dimension D(400,400),X(400,20),BW(400,400),W(160000),BB(160000)
      DIMENSION Y(400,20)
      dimension A(400),B(400),H(400),P(400),Q(400),TITLE(20),FMT(6)
      dimension DLO(80000),DHI(80000)
      integer BW,A,B,P,Q,QIC,QJC,N,Z,Y
      data PI,EPS,ZD,ZX/3.14159,1.0e-6,'d','x'/

c Les variables seront expliquees au fur et a mesure.

c*****************************************************************

      open(unit=30,file='deubs.dat',status='old')
      open(unit=40,file='memoireres.dat',status='new',err=5)
      goto 6
 5    open(unit=40,file='memoireres.dat',status='old')
 6    continue
c Les fichiers deubs.dat (et memoireres.dat) contiennent (contiendront)
c   evidemment les donnes a anlyser (et les resultats).
```

```
c*********************************************************************

c Lecture des donnees:
c----------------------

      read(30,506) (TITLE(J),J=1,20)
      write(40,506) (TITLE(J),J=1,20)

c On lit la premiere ligne du fichier de donnees qui doit comporter
c    son nom.

      read(30,503) F

c On lit la deuxieme ligne du fichier de donnees qui comporte un "d" si
c   les donnees sont sous la forme d'une matrice de dissimilarite
c   (partie triangulaire inferieure de cette matrice), et
c   un "x" si elles sont sous la forme d'une "pattern matrix"
c     (une ligne=un objet ; une colonne=une variable caracteristique).

      if (F.eq.ZD) go to 71

c Si la matrice donnee est la matrice de dissimilarite, on passe a 71.
c Sinon on va la calculer :

      read(30,*) N,NP,NHI
      write(40,605) N,NP
      write(*,*) N,NP,nhi
      FP=float(NP)

c    -on commence par lire le nombre de donnees ("N"), le nombre de
c      variables ("NP", puis "FP"), et le nombre de groupes ("NHI") pour
c      lequel on veut afficher la partition obtenue (une serie de nombres
c      indiquant a quel groupe appartient le premier objet, le deuxieme,
c      etc).

      do 72 i=1,N
         read(30,502) (X(I,K),K=1,NP)
72       write(*,*) (X(I,K),K=1,NP)

      DO I=1,3
      DO J=1,NP
      write(*,*) X(I,J)
      enddo
      enddo
```

```
c     -on lit ensuite la "pattern matrix" (ne pas oublier d'adapter
c         le format).

      call xtod(N,NP,X,D)
      go to 73

c     -on calcule la matrice de dissimilarite ("D=(D(I,J)") grace a la
c         sous-routine "xtod", et on va en 73.
c         Cette matrice est symetrique et a des 0 sur la diagonale.
c    ATTENTION: D(I,J) = distance entre I et J.

c On arrive alors en 71 pour le cas ou la matrice de dissimilarite
c     etait donnee. Dans ce cas,

71    read(30,501) N,NHI
      write(40,607) N

c     -on lit le nombre de donnees ("N"), on l'ecrit dans le fichier
c       resultat et on lit le nombre de groupes pour lequel
c       on veut afficher la partition obtenue ("NHI").

      do 74 i=2,N
      I1=I-1
      read(30,fmt) (d(i,j),j=1,i1)
      do 74 j=1,i1
74    d(j,i)=d(i,j)

c     -on lit cette matrice de dissimilarite ("D=(D(I,J)").

73    continue
**********************************************************************

c Programme:
c------------

      do 121 nclust=1,4

c Chaque valeur de "nclust" represente une methode de classification.
c Donc, pour chacune des quatres methodes, on va faire ce qui suit.

      do 75 i=2,n
      i1=i-1
      do 75 j=1,i1
75    bw(i,j)=0
```

```
c On remplit la partie de la matrice "BW" sous la diagonale avec des
c    zeros.
c On verra apres ce que represente cette matrice.

      go to (81,82,83,84) nclust
81    write(40,611)
      go to 90
82    write(40,612)
      go to 90
83    write(40,613)
      go to 90
84    write(40,614)
90    write(40,608)

c On ecrit le nom de la methode de classification utilisee dans le
c   fichier resultat.

      if(f.eq.zd) write(40,609)
      write(40,610)

c On indique dans le fichier resultat quelle etait la matrice donnee.
c Ensuite, on ecrit les titres des colonnes qui vont contenir les
c    resultats (voir format 610).

      ttot=0.0
      k=0

      do 101 i=1,n-1
      do 101 j=i+1,n
      k=k+1
      dlo(k)=d(j,i)
101   ttot=ttot+d(j,i)

      ttot=ttot/float(n)

c On calcule le tableau "DLO" qui contient PROVISOIREMENT les
c     dissimilarites de la matrice triangulaire inferieure "D=(D(I,J))"
c     prises colonne par colonne.
c De plus, on calcule la moyenne de ces dissimilarites "TTOT".

      nc2=n*(n-1)/2
      call badsrt(dlo,dhi,nc2)

c Soit "NC2", le nombre d'elements de "DLO".
c On trie "DLO" grace a la sous-routine "badsrt".
```

```
c DONC, "DLO" contient toutes les dissimilarites triees par
c    ordre CROISSANT.

      wtot=0.0
      dtot=0.0
      ndtot=0
      hold=0.0
      n1=n-1
      k=0

      do 1 i=1,n
      p(i)=i
1     q(i)=1

c On initialise "P" par [1,2,...,N] et "Q" par [1,1,...,1].
c "P" contient des valeurs qui sont les memes (ex: P[2]=2 et P[4]=2)
c    si les points correspondants sont dans la meme classe (les 2eme
c    et 4eme points sont dans la meme classe).
c    ATTENTION: le premier point d'une classe a toujours sa valeur
c    dans "P" (P[2]=2).
c "Q" est lie a "P". Il contient, quand plusieurs points sont dans la
c     meme classe, le nombre de points dans cette classes. Seulement,
c     ce nombre est indique dans la case correspondant au
c     premier de ces elements (dans l'exemple, Q[2]=2 et Q[4]=1).

C++++++++++++++++++++++++++++++++++++++++++++++++
c Tant que K est different de N1=N-1, on fait:
c  (car quand K=N1, a un moment, on a "goto 22" qui fait passer
c   au dessus de l'instruction "goto 2").

2     k=k+1
      dmax=1.0e20

c Partir de maintenant, on va travailler sur la partie superieure
c   de "D=(D(I,J))". Donc "D"=PARTIE SUPERIEURE DE "D" POUR APRES.
c   Cette partie va contenir les distances entre les
c   points ou les groupes de points formes. En fait, c'est la matrice
c   que l'on transforme habituellement au fur et a mesure des etapes
c   (et donc des groupements ou des divisions)pour les methodes
c   hierarchiques.

      do 4 i=1,n1
      if (p(i).ne.i) go to 4
      i1=i+1
      do 3 j=i1,n
```

```
      if (p(j).ne.j) go to 3
      t=d(i,j)
      if (t.gt.dmax) go to 3
      ic=i
      jc=j
      dmax=t
3     continue
4     continue

c On recherche le plus petit element de "D=(D(I,J))" actuel, SAUF
c    parmi ceux regroupes avant (c'est pourquoi on inspecte les
c    valeurs de "P=(P(I))". En fait, on ne regarde que les distances
c    entre les groupes deja formes.
c Cet element est note "DMAX". Il se trouve a la ligne "IC" et a la
c    colonne "JC" dans "D". Ce sont alors les deux classes comprenant
c    "IC" et "JC" respectivement que l'on va regrouper.

      call wgss(d,p,q,n,ic,ssq1,dw1)

c On calcule la somme "DW1" et la moyenne "SSQ1" des dissimilarites
c   des elements appartenant a la meme classe que "IC".

      call wgss(d,p,q,n,jc,ssq2,dw2)

c On calcule la somme "DW2" et la moyenne "SSQ2" des dissimilarites
c   des elements appartenant a la meme classe que "JC".

      dwt=dw1+dw2
      nw1=(q(ic)*(q(ic)-1))/2
      nw2=(q(jc)*(q(jc)-1))/2
      nwt=nw1+nw2
      ssqt=ssq1+ssq2
C      write(*,*)'ssqt',ssqt
C      write(*,*)'ssq1,ssq2,',SSQ1,SSQ2
c On calcule:
c      -la somme sur les deux classes regroupees des dissimilarites
c        "intra-classes" "DWT".
c      -la somme sur les deux classes regroupees des moyennes des
c        dissimilarites "intra-classes" "SSQT".
c      -la somme sur les deux classes regroupees des nombres de D(I,J)
c        par classes (4 objets -> 6 D(I,J)).

      do 21 j=1,n
      if (p(j).ne.jc) go to 21
      do 23 i=1,n
```

```
      if (p(i).ne.ic) go to 23
      ii=max0(i,j)
      jj=min0(i,j)
      bw(ii,jj)=1
23    continue
      p(j)=ic
21    continue
      a(k)=ic
      b(k)=jc
      h(k)=dmax
      qic=q(ic)
      qjc=q(jc)
      z=qic+qjc
      ff=1.0/float(z)

      do 20 i=1,n
      if (i.eq.ic.or.p(i).ne.i) go to 20
      if (nclust.ne.3) go to 27
      qi=q(i)
      z=qi+qic+qjc
      ff=1.0/float(z)
27    j=min0(ic,i)
      l=max0(ic,i)
      k1=min0(jc,i)
      k2=max0(jc,i)
      dj=d(j,l)
      dk=d(k1,k2)
      go to (111,112,113,114) nclust
c     SINGLE LINK
111   d(j,l)=amin1(dj,dk)
      go to 20
c     GROUP AVERAGE LINK
112   d(j,l)=ff*(qic*dj+qjc*dk)
      go to 20
c     WARD'S METHOD
113   d(j,l)=ff*((qic+qi)*dj+(qjc+qi)*dk-qi*dmax)
      go to 20
c     COMPLETE LINK
114   d(j,l)=amax1(dj,dk)
20    continue
      q(ic)=q(ic)+q(jc)
      call wgss(d,p,q,n,ic,ssq,dw)
      nqc=q(ic)
      nw=(nqc*(nqc-1))/2
      fq=float(nqc)
```

```
      if (f.eq.zd) go to 93
      if (ssqt-eps) 93,93,94
94    beale=(ssq-ssqt)/ssqt
      p2=2.0/fp
      div=((fq-1.0)/(fq-2.0))*(2.0**p2)-1.0
      beale=beale/div
      go to 95
93    beale=0.0
95    continue
      if (f.eq.zd) go to 92
      hold=ssqt/ssq-1.0+2.0/(pi*fp)
      hold2=(2.0-16.0/(pi*pi*fp))/(fp*fq)
      hold=-hold/sqrt(hold2)
92    wtot=wtot+ssq-ssqt
      btot=ttot-wtot
      if (k.eq.n1) go to 22
      z=n-k-1
      ch=btot*float(k)/(wtot*float(z))
      dtot=dtot+dw-dwt
      ndtot=ndtot+nw-nwt
      cind=(dtot-dlo(ndtot))/dhi(ndtot)
      kin=0
      kout=0
      do 31 i=2,n
      i1=i-1
      do 31 j=1,i1
      if (bw(i,j)) 32,32,33
32    kout=kout+1
      bb(kout)=d(i,j)
      go to 31
33    kin=kin+1
      w(kin)=d(i,j)
31    continue
      u=0.0
      dd=0.0
      do 51 kk=1,kin
      din=w(kk)
      do 51 jj=1,kout
      if (din-bb(jj)) 51,52,53
52    dd=dd+0.5
      go to 51
53    u=u+1.0
51    continue
      denmr=float(kin)*float(kout)/2.0-dd
      gamma=1.0-u/denmr
```

```
      ngp=n-k
      write(40,601) ngp,h(k),a(k),b(k),gamma,hold,beale,ch,cind
      if (ngp-12) 41,42,41
41    if (ngp-nhi) 2,42,2
42    write(40,602) (p(ll),ll=1,n)
      go to 2
C+++++++++++++++++++++++++++++++++++++

22    ngp=n-k
      write(40,616) ngp,h(k),a(k),b(k),hold,beale
121   continue
501   format(6i4)
502   format(f3.1,1x,f3.1,1x,f3.1,1x,f3.1)
503   format(a1)
506   format(20a4)
601   format(i4,f10.3,2i4,2x,f8.6,1x,f9.5,1x,f9.5,1x,f10.4,1x,f8.6)
602   format(15i4)
603   format(10f8.1)
604   format(10f8.5)
605   format('data provided as pattern matrix',/,'number of objects=',
     1   i4,'  number of dimensions=',i4,/)
606   format(i4,10f8.4/(4x,10f8.4))
607   format('data provided as dissimilarity matrix',/,
     1  'number of objects=',i4,/)
608   format(/,' evaluation of partition gamma, duda-heart, beale,'
     1  ' calinski-harabasz',/,'   and c statistics',/)
609   format(/,' data not allow evaluation of duda-heart and beale'
     1  ' statistics')
610   format(/,' ngps   height  leaders',
     1  '   gamma   duda-heart  beale  c-h         c')
611   format(/,' single link analysis')
612   format(/,' group average link analysis')
613   format(/,' wards method analysis')
614   format(/,' complete link analysis')
616   format(i4,f10.3,2i4,11x,f9.5,1x,f9.5)
      close(30)
      close(40)
      stop
      end

      subroutine xtod(n,np,x,d)
      dimension x(400,20),d(400,400)
      do 1 i=2,n
      i1=i-1
      do 3 j=1,i1
```

```
      tot=0.0
      do 2 k=1,np
      z=x(i,k)-x(j,k)
2     tot=tot+z*z
      d(i,j)=tot
3     d(j,i)=d(i,j)
1     continue
      return
      end

      subroutine wgss(d,p,q,n,ic,ssq,dw)
      dimension d(400,400),p(400),q(400)
      integer p,q
      ssq=0.0
      dw=0.0
      if (q(ic).le.1) return
      do 1 i=1,n-1
      if (p(i).ne.ic) go to 1
      do 2 j=i+1,n
      if (p(j).ne.ic) go to 2
      ssq=ssq+d(j,i)
2     continue
1     continue
      dw=ssq
      ssq=ssq/q(ic)
      return
      end

      subroutine badsrt(dlo,dhi,nc2)
      dimension dlo(45000),dhi(45000)
      do 1 i=1,nc2-1
      small=dlo(i)
      do 1 j=i+1,nc2
      if (small-dlo(j)) 1,1,2
2     dlo(i)=dlo(j)
      dlo(j)=small
      small=dlo(i)
1     continue
      dhi(1)=dlo(nc2)
      do 4 i=2,nc2
4     dhi(i)=dhi(i-1)+dlo(nc2-i+1)
      do 5 i=2,nc2
5     dlo(i)=dlo(i-1)+dlo(i)
      do 3 i=1,nc2
3     dhi(i)=dhi(i)-dlo(i)
```

```
return
end
```