LES CAHIERS DE L'ANALYSE DES DONNÉES

C. ROSENZVEIG

Une chaîne d'analyse des correspondances sur micro-ordinateur

Les cahiers de l'analyse des données, tome 3, n° 4 (1978), p. 418-434

<http://www.numdam.org/item?id=CAD_1978__3_4_418_0>

Les Cahiers de l'Analyse des Données
Vol. III - 1978 - n° 4 - p. 418-434

UNE CHAÎNE D'ANALYSE DES CORRESPONDANCES SUR MICRO-ORDINATEUR [CORRESPONDANCES MICRO-ORDINATEUR]

par C. Rosenzveig (1)

1 *Introduction*

Ayant acquis en Juin 1976 un micro-ordinateur (HEWLETT-PACKARD type 9825A) dont la capacité de mémoire centrale est de 23 kilo-octets nous y avons transféré un fichier concernant les réponses de 5756 sous-officiers à 186 questions (2). En raison de contraintes techniques (absence de lecteur de bande magnétique) nous avons utilisé comme support intermédiaire la bande perforée qui nous a permis d'aboutir à l'utilisation actuelle de mini-cassettes magnétiques.

Une chaîne de traitement complet encombrant au minimum la mémoire centrale de ce micro-ordinateur, nous permet d'analyser par exemple 70 questions dichotomiques ou 14 questions à 6 items. Le temps de lecture et de calcul pour 5756 individus décrits par 70 modalités logiques est avec cette chaîne d'environ 6 h 30 (3 h 30 + 3 h) . L'utilisation d'un disque souple, en option, diminuerait beaucoup le temps de lecture.

Les programmes sont décrits ci-dessous, en particulier une construction de base orthonormée en vecteurs n'ayant que deux composantes distinctes (en omettant les composantes nulles ou identiques) nous a été très utile, nous la donnons au § 4.

2 *Méthode de programmation*

Ce micro-ordinateur portable (12 kg) dont un exemplaire se trouve à la Maison des Sciences de l'Homme (deuxième sous-sol, salle 209) utilise un langage très proche du L.S.E. (Langage Symbolique d'Enseignement) bien connu dans nos lycées et universités (MITRA 15).

Les calculs sont systématiquement effectués sur 12 décimales, soit 8 octets (double précision). Grâce aux chaînes de caractères il est possible d'accéder :

- au mot de 4 octets (précision IBM) en utilisant la fonction de codage "fts" (*float to simple*) l'opération inverse étant effectuée par la fonction "stf".
- à l'entier de 2 octets ($-32760 \leq N \leq 32767$) en utilisant la fonction de codage "fti" (*float to integer*) l'opération inverse étant effectuée par la fonction "itf".
- à l'octet ($0 \leq N \leq 255$) en utilisant la fonction de codage " char " (caractère) l'opération inverse étant effectuée par la fonction "num".
- au bit grâce à la fonction "bit".

Le recours à ces fonctions alourdit la programmation mais permet

(1) *Docteur 3° cycle. Assistant en Sciences économiques à Paris X (Nanterre).*

(2) *Ce fichier nous a été fourni par le Centre de Sociologie de la Défense Nationale dirigé par M. J.P.H. Thomas.*

de gagner beaucoup de place : il n'est pas très économique par exemple de ranger la réponse à une question dichotomique (0 ou 1) sur 8 octets!

Celles-ci nous ont permis de diagonaliser des matrices symétriques d'ordre 72 (méthode du QL implicite).

La multiplicité des fonctions pré-programmées par le constructeur (calcul matriciel, logarithme, exponentielle, minimum, modulo, génération de nombres au hasard,...) en font un outil très accessible au non-spécialiste.

Faisons une remarque en ce qui concerne l'exécution des boucles. Voici un programme calculant la somme des N premiers nombres pairs :

```
0 → S ; for I = 2 to N by 2 ; S + I → S ; next I
```

On remarque déjà que plusieurs instructions peuvent être écrites sur une même ligne. Il suffit de les séparer par un point virgule. L'affectation est symbolisée par une flèche. Le début de la boucle est marqué par "for". L'indice étant I, la fin de la boucle est délimitée par "next I". Le pas symbolisé par "by" est 2. A la différence du FORTRAN la boucle n'est pas systématiquement parcourue au moins une fois car le test sur I est effectué au début de la boucle. Si N par exemple est égal à 0 la boucle n'est pas effectuée donc S vaut toujours 0 (I = 2).

En rédigeant ces programmes, j'ai recherché la compacité pour des raisons évidentes d'économie de temps de calcul et de mémoire centrale, ce qui explique qu'ils soient difficiles à lire. C'est pourquoi le listing de la chaîne de traitement (5000 instructions environ) ne figure pas ici ; cependant je me tiens à la disposition de tout lecteur désireux de les obtenir.

Ce qui a rendu cette programmation délicate fut le souci permanent de diagonaliser la matrice d'ordre (card J - Card Q) le plus grand possible. Par conséquent il n'était pas possible de créer le tableau de Burt en entier d'ordre card J x card J mais seulement un tableau d'ordre (card J - card Q) x (card J - card Q) appelé Burt incomplet où dans chaque bloc Jq x Jq', sont supprimées la dernière ligne et la dernière colonne que l'on peut reconstituer grâce au tableau des poids des items-colonnes. Bien sûr à aucun moment le tableau logique des réponses, trop volumineux, n'est créé en mémoire centrale!

Les principales difficultés sont apparues :

- à la lecture des données en vue de la création du tableau de Burt incomplet.
- l'impression du triangle inférieur complet du tableau de Burt nécessairement reconstitué bloc par bloc.
- au compactage par élimination des éventuelles lignes et colonnes de poids nuls.
- surtout dans le calcul par bloc de ^{t}UBU (où B est le tableau de Burt incomplet et U une matrice compacte n'ayant que 2 lignes (cf § 4).

Cette chaîne de traitement composée de 21 programmes peut paraître trop longue. Cependant les praticiens savent d'expérience qu'il est plus facile de grouper des programmes en les mettant bout à bout et en supprimant les appels, devenus inutiles, de certains fichiers temporaires que d'en faire éclater un trop encombrant puis de créer ces mêmes fichiers.

3 Description des programmes

- BURT 01 - Il crée un dictionnaire dans la chaîne de caractère Z$ où sont écrits 186 blocs de 7 octets à raison d'un bloc par question, et de 16 octets utilisés pour des besoins internes au système ("buffer").

Ces 7 octets se décomposent comme suit :

- 4 octets repèrent le code de la question. Exemple : 147d.

- le 5° repère le nombre des réponses possibles (y compris les non-réponses).

- le 6° octet repère l'emplacement physique dans le tableau des données de la colonne - début de la réponse à une question.

- le 7° octet repère la colonne - fin de la réponse à cette question.

- BURT 02 - Il permet de sélectionner un sous-ensemble de questions sans utiliser l'instruction FORMAT.

Le programme vérifie si le code de la question demandée est correct et, dans ce cas, demande s'il faut regrouper certains items (par exemple : non réponse avec 1) afin d'éviter les colonnes de poids trop faible. Plusieurs tests de cohérence sont effectués sur la fonction de regroupement. Presser la touche "clear", après un test négatif, annule la question en cours. Seuls les noms, des items associés aux points fixes de la fonction de regroupement, sont conservés. Un test vérifie qu'il reste au moins deux items par question. Signalons que l'ordre d'introduction des questions n'a aucune importance sur les résultats. Le nom des questions, l'étendue de leurs items et la fonction de regroupement éventuelle sont imprimés au fur et à mesure de leur introduction, après vérification des tests.

La réponse 0 à la question : "Effectif analysé ?" signifie que tout l'échantillon est pris en compte. Il crée aussi une table d'adresses des réponses dans le tableau de Burt incomplet.

- BURT 03 - Il initialise à 0 la matrice de Burt incomplète et le tableau des effectifs de chaque item-colonne. Il crée aussi les titres des cassettes-données à utiliser. La chaîne de caractère A$ sert à entrer les données, enregistrement par enregistrement, correspondant à 6 cartes perforées concernant les réponses des individus groupés deux par deux.

- BURT 04 - Il lit enregistrement par enregistrement le tableau des données puis sélectionne les questions et calcule le triangle *supérieur* de la matrice de Burt incomplète ainsi que les poids des items à l'aide du dictionnaire créé dans Burt 01 et teste le nombre d'individus lus.

- BURT 05 - Il imprime le nom de chaque question retenue ainsi que le nombre de ses items (s'il y a eu regroupement, il s'agit des *nouveaux* items) en vérifiant que la somme de leurs poids est bien égale à l'effectif analysé. Il décide l'exécution ou non de BURT 07.

- BURT 06 - Edite la totalité du triangle *inférieur* du tableau de Burt complet après l'avoir reconstitué bloc par bloc à l'aide du tableau des poids des items (dans la mémoire centrale, c'est le triangle *supérieur* qui est calculé).

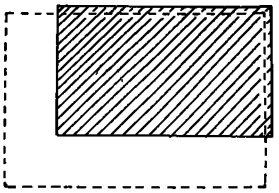
- BURT 07 - Recherche les items de poids nul ou égal à l'effectif total de l'échantillon. Pour chaque item :

- si le poids est nul, il élimine la ligne et la colonne correspondante dans la matrice de BURT incomplète.

- si le poids est égal à l'effectif total, toutes les lignes et colonnes de la question correspondant à cet item sont éliminées. Il vérifie si le nombre d'items est supérieur à 0. Lors du calcul de BURT 16 , ces items de coordonnées nulles sont réintroduits à leur place.

- BURT 08 - Calcule la base k_J orthonormale (et non f_J pour des questions de précision (cf remarque 2 § 4.3).

- BURT 09 - Ce programme d'élaboration très délicate mérite un commentaire moins succinct. Il calcule ${}^tUBU$ par blocs. Ceux-ci sont pris dans le triangle *supérieur* et, après calcul, les nouveaux blocs sont placés dans le triangle *inférieur* de la matrice de BURT incomplète. Cependant il faut reconstituer les blocs incomplets en leur adjoignant la dernière ligne et la dernière colonne reconstituées à partir du tableau des poids des items, ce que nous avons fait en décalant les éléments (cf figure). Nous y reviendrons.



La partie en tirets représente le bloc reconstitué débordant d'une colonne à gauche et d'une ligne au-dessous.

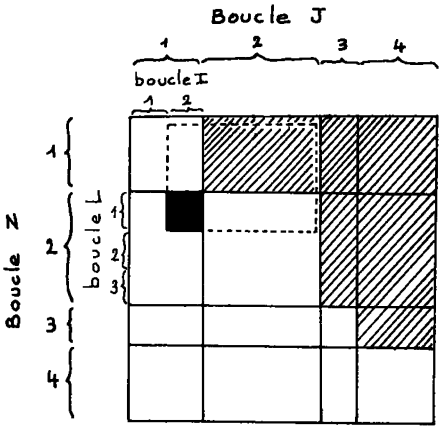
Etant donné que les blocs (q,q) de la matrice à diagonaliser ${}^tUBU$ sont des matrices identités à une constante multiplicative près, le calcul se fait sur des blocs non diagonaux, ce qui permet dans tous les cas ce décalage (même dans le cas de questions dichotomiques). Malheureusement ceci implique plusieurs contraintes au niveau du sens de parcours des boucles.

Exemple : Dans la figure suivante quatre questions sont analysées. Afin de ne pas détruire l'information figurant dans les blocs hachurés, il est nécessaire :

- soit de faire croître J, indice colonne des blocs, de 2 à 4 puis pour J fixé de faire décroître Z, indice ligne des blocs, de J - 1 à 1.

- soit de faire décroître Z de 3 à 1 puis, pour Z fixé, de faire croître J de Z + 1 à 4 (inversion dans l'imbrication des boucles).

Nous avons choisi la première solution en raison du calcul de la fonction "modulo" (cf programme : exemple de calcul de BU § 4.4). D'autre part, le bloc reconstitué (1 x 2) recoupe sur la case noire le bloc à créer (2 x 1). Nous avons donc à nouveau le choix entre deux solutions afin de terminer le calcul par la détermination de cette case :



Ces indices sont utilisés dans le programme BURT09 de l'annexe.

Nous avons choisi la première solution en raison du calcul de la fonction "modulo" (cf programme : exemple de calcul de BU § 4.4). D'autre part, le bloc reconstitué (1 x 2) recoupe sur la case noire le bloc à créer (2 x 1). Nous avons donc à nouveau le choix entre deux solutions afin de terminer le calcul par la détermination de cette case :

- soit de faire décroître l'indice L de 3 à 1 puis croître I de 1 à 2
- soit de faire croître I de 1 à 2 puis décroître L de 3 à 1.

Bien que la deuxième solution paraisse moins intéressante (boucle décroissante incluse dans une boucle croissante) elle est plus rapide car la boucle décroissante ne fait pas appel au calcul de la fonction "minimum" (cf même programme que précédemment). Afin de calculer les quantités A, B, C, D (cf § 4.4) il faut inclure deux nouvelles boucles, ce qui porte à 6 le niveau d'imbrication. Cependant il est possible d'éviter le 6° lors de la reconstitution des blocs. En effet, examinons le bloc incomplet (1x2) :

BLOC PRIMITIF

B11	B12	B13
B21	B22	B23

BLOC RECONSTITUE

B11	B11 + B12	B11 + B12 + B13	k_1
B21	B21 + B22	B21 + B22 + B23	k_2
B31	B31 + B32	B31 + B32 + B33	k_3

Le calcul des sommes partielles ci-dessus dès la reconstitution du bloc évite de le faire au niveau de la 6° boucle imbriquée.

- BURT 10, 11, 12 - Diagonalisation de la matrice ${}^tUBU$ par la méthode du QL implicite. On obtient toutes les valeurs propres et tous les vecteurs propres orthonormés pour la métrique unité.

- BURT 13 - Transposition de la matrice des vecteurs propres qui, pour des raisons techniques dues à l'utilisation de tableau de chaîne de caractères, permet d'accélérer la procédure de tri utilisée dans BURT 14.

- BURT 14 - L'algorithme de tri très rapide et d'encombrement très faible (4 lignes) est issu de l'algorithme de SHELL. Nous avons amélioré sa rapidité. Examinons l'exemple suivant : on veut ranger en ordre croissant la suite suivante :

	1	2	3	4	5	6	7
S =	1	1	1	1	1	1	0

L'algorithme (cf exemple : lignes 4 à 8) permute les contenus de S[7] et S[4] et de S[4] et S[1] donc effectue 6 affectations :

S[7]→S ; S[4]→S[7] ; S→S[4]

puis S[4]→S ; S[1]→S[4] ; S→S[1]

il est plus simple de faire :

S[7]→S ; S[4]→S[7] ; S[1]→S[4] ; S→S[1]

```
0: "Le symole ) represente l'assignation";
1: "1)A signifie A=1 en FORTRAN";
2: "int(M)= partie entiere de M";
3: "Multiplication implicite : .5M=.5*M ";
4: 7)N)M
5: gto +4;if int(.5M))M;for J=M+1 to N;S(J)K))S
6: if S(S(K-M)L);S(L))S(K);jmp (L)K)(=M
7: S)S(K)
8: next J;gto -3
9: end
```

évitant ainsi les affectations inutiles : S → S[4] et S[4] → S.

Les vecteurs propres sont rangés simultanément dans le même ordre que les valeurs propres.

- BURT 15 - On calcule les coordonnées des items conservés après le compactage effectué dans BURT 07. Ces coordonnées (10 au maximum) sont rangées à raison d'une fonction coordonnée par fichier.

- BURT 16 - Réintroduit à leur bonne place, s'il est appelé, les coordonnées nulles des items de poids nuls et les items des questions à réponses unanimes provisoirement écartés.

- BURT 17 - Imprime les histogrammes classiques des valeurs propres du tableau logique et de leurs carrés (tableau de Burt) ainsi que les taux et cumuls. Un tableau dont le nombre de colonnes s'ajuste automatiquement au nombre N (compris entre 1 et 10) des facteurs contient en ligne pour chaque item son nom, le nombre des individus qui l'ont choisi, ses N coordonnées, ses N contributions absolues, ses N contributions relatives et enfin, en dernière colonne, sa variance totale par rapport au centre de gravité.

- BURT 18 - A peu près analogue à BURT 02. Sélectionne les questions supplémentaires sans regrouper les items car il est plus informatif de faire apparaître sur le graphique des chapelets de points dont la disposition aura une signification certaine malgré le faible poids de certains d'entre eux que de placer quelques points très lourds, généralement situés près du centre de gravité, estompant ainsi l'effet de forme. Il crée aussi une table de pointeurs des questions dans le tableau des données.

- BURT 19 - Calcule les coordonnées des items supplémentaires . Comme il n'est pas possible d'entrer en mémoire centrale les N($\leq$10) coordonnées de tous les individus (5756 x 10 dans notre cas), il est nécessaire de les calculer au fur et à mesure, individu par individu. On calcule aussi sur les N axes les variances des groupes I_s d'individus ayant choisi l'item s.

- BURT 20 - Imprime le tableau de dimension variable (cf BURT 17) comprenant pour chaque item supplémentaire : son nom, le nombre des individus l'ayant choisi, ses N coordonnées, ses N contributions relatives, les N variances du groupe I_s, et sa variance totale par rapport au centre de gravité des items ayant participé à l'analyse.

- GRAPHE - Représente à l'intérieur d'un cadre d'impression, dont on fournit le nombre de colonnes, un système d'axes orthonormés dont les graduations s'éloignent symétriquement de l'origine, y projette des points identifiés par 225 caractères au plus dont celui u milieu (dans le cas impair) ou le plus proche à gauche du milieu (dans le cas pair), repère l'abscisse. Les points hors cadre y sont ramenés en option. Grâce à un seul tri les identificateurs sont rangés dans l'ordre d'écriture d'une imprimante classique (gauche à droite et de bas en haut). Les parties des identificateurs qui se confondent contiennent des astérisques et les identificateurs complets apparaissent regroupés entre des parenthèses sous le graphe. Beaucoup de commentaires figurent dans le programme. Signalons aussi la possibilité de ne représenter qu'un seul axe, soit verticalement soit horizontalement, le cadre graphique étant bien entendu supprimé.

4 . Construction d'une base f-orthonormée d'encombrement minimal

Soit un ensemble fini J de n($\geq$2) points j affectés des masses f_j strictement positives et de somme quelconque. Sur $\mathbb{R}^J$ se trouve défini le produit scalaire :

$$\forall\ U^J\ ,\ V^J \in \mathbb{R}^J\ :\ <U^J\ ,\ V^J>_f\ =\ \Sigma\{f_j\ U^j\ V^j \mid j \in J\}$$

auquel est associée la norme : $\|U^J\|_f\ =\ (<U^J\ ,\ U^J>_f)^{1/2}$

Soit une base (U^J_k) $(1 \leq k \leq n)$ de $\mathbb{R}^J$ où nous imposons à la dernière fonction U^J_n d'être constante sur J (pour des raisons qui apparaîtront plus loin). Dans la suite nous omettrons souvent l'indice supérieur J ainsi que l'indice inférieur f du produit scalaire et de la norme.

4.1 Rappel de la méthode d'Hilbert-Schmidt

En faisant décroître l'indice k de n à 1 on construit par récurrence la base f-orthonormée (U''_k) :

$$U'_n\ =\ U_n \qquad U''_n\ =\ \frac{U'_n}{\|U'_n\|}$$

$$\forall\ k \in \{1, \ldots,\ n-1\}$$

$$U'_k\ =\ U_k - \Sigma\{<U_k\ ,\ U''_\ell>\ U''_\ell \mid k < \ell \leq n\} \qquad U''_k\ =\ \frac{U'_k}{\|U'_k\|}$$

En bref pour $1 \leq k \leq n-1$ on ne retient successivement de chaque U_k que la composante normale f-unitaire au sous-espace engendré par les U''_ℓ pour $k < \ell \leq n$. (Ce sous-espace est aussi engendré par les U_ℓ pour $k < \ell \leq n$). Cette méthode, valable quelle que soit la métrique définie positive sur $\mathbb{R}^J$, présente deux inconvénients :

- elle est coûteuse en temps de calcul car pour chaque fonction U_k il est nécessaire de calculer $n-k+1$ produits scalaires dont la racine carrée du dernier sert à calculer une norme.

- sa précision diminue avec n car le calcul par récurrence des composantes conduit au cumul des erreurs dues aux troncatures et arrondis qui ont peu de chances de se compenser.

La métrique diagonale définie positive sur $\mathbb{R}^J$ utilisée en analyse des correspondances nous permet de construire une base f-orthonormée se terminant par une fonction constante et dont toutes les composantes non-nulles distinctes pourront être conservées dans un tableau à deux lignes et n colonnes. Prenons l'exemple suivant où n = 5.

$$\begin{array}{ccccc}
U_1 & U_2 & U_3 & U_4 & U_5 \\
\left(\frac{f_2}{f_1(f_1+f_2)}\right)^{1/2} & \left(\frac{f_3+f_4}{(f_1+f_2)(f_1+\ldots+f_4)}\right)^{1/2} & 0 & \left(\frac{f_5}{(f_1+\ldots+f_4)(f_1+\ldots+f_5)}\right)^{1/2} & c \\
-\left(\frac{f_1}{f_2(f_1+f_2)}\right)^{1/2} & \left(\frac{f_3+f_4}{(f_1+f_2)(f_1+\ldots+f_4)}\right)^{1/2} & 0 & \left(\frac{f_5}{(f_1+\ldots+f_4)(f_1+\ldots+f_5)}\right)^{1/2} & c \\
0 & -\left(\frac{f_1+f_2}{(f_3+f_4)(f_1+\ldots+f_4)}\right)^{1/2} & \left(\frac{f_4}{f_3(f_3+f_4)}\right)^{1/2} & \left(\frac{f_5}{(f_1+\ldots+f_4)(f_1+\ldots+f_5)}\right)^{1/2} & c \\
0 & -\left(\frac{f_1+f_2}{(f_3+f_4)(f_1+\ldots+f_4)}\right)^{1/2} & -\left(\frac{f_3}{f_4(f_3+f_4)}\right)^{1/2} & \left(\frac{f_5}{(f_1+\ldots+f_4)(f_1+\ldots+f_5)}\right)^{1/2} & c \\
0 & 0 & 0 & -\left(\frac{f_1+f_2+f_3+f_4}{f_5(f_1+f_2+f_3+f_4+f_5)}\right)^{1/2} & c
\end{array}$$

où $c\ =\ (f_1 + f_2 + \ldots + f_5)^{-1/2}$

Il est facile de voir en éliminant les composantes nulles ou redondantes qu'il n'en reste que 2 x 4 (+1 pour le dernier vecteur) !

4.2 Construction d'un système centré orthogonal

En construisant un système orthogonal de n - 1 fonctions centrées (i.e. de moyenne nulle sur J) puis en ajoutant une fonction constante U_n sur J on obtient une base orthogonale sur J. En effet :

$$\forall\ 1 \le k \le n-1 : \langle U_n , U_k \rangle = \Sigma\{f_j\ U_n^j\ U_k^j \mid 1 \le j \le n\}$$

$$= U_n^1\ \Sigma\{f_j\ U_k^j \mid 1 \le j \le n\} = 0$$

car les U_k sont centrées.

Pour construire le système orthogonal nous distinguerons deux cas:

A - le nombre n de composantes est pair (≥2)

si n = 2 la fonction $\begin{bmatrix} f_2 \\ -f_1 \end{bmatrix}$ est bien centrée.

Supposons construite la matrice du système orthogonal d'ordre impair formé de n-1 fonctions centrées à n composantes et bordons cette matrice par les deux lignes et les deux colonnes suivantes :

	$U_1 \ldots U_{n-1}$	U_n	U_{n+1}
1	Système orthogonal de fonctions centrées	$f_{n+1} + f_{n+2}$	0
.		"	"
.		:	:
.		"	"
n		$f_{n+1} + f_{n+2}$	0
n + 1	0 0	$-\Sigma\{f_j \mid 1 \le j \le n\}$	f_{n+2}
n + 2	0 0	$-\Sigma\{f_j \mid 1 \le j \le n\}$	$-f_{n+1}$

Les colonnes U_1 , ..., U_{n-1} restent évidemment orthogonales et centrées puisque leur n+1 et n+2 ièmes composantes sont nulles. Montrons que U_n et U_{n+1} sont centrées:

a) $\Sigma\{f_j\ U_n^j \mid 1 \le j \le n+2\}$

$= \Sigma\{f_j (f_{n+1} + f_{n+2}) \mid 1 \le j \le n\} - f_{n+1}\Sigma\{f_j \mid 1 \le j \le n\} - f_{n+2}\Sigma\{f_j \mid 1 \le j \le n\}$

$= (f_{n+1} + f_{n+2})\Sigma\{f_j \mid 1 \le j \le n\} - (f_{n+1} + f_{n+2})\Sigma\{f_j \mid 1 \le j \le n\} = 0$

b) $\Sigma\{f_j\ U_{n+1}^j \mid 1 \le j \le n+2\} = f_{n+1}\ f_{n+2} - f_{n+2}\ f_{n+1} = 0$

Montrons maintenant l'orthogonalité :

$\forall\ k \in \{1, \ldots, n-1\}$:

a) $\langle U_n , U_k \rangle = \Sigma\{f_j\ U_n^j\ U_k^j \mid 1 \le j \le n+2\}$

$= \Sigma\{f_j (f_{n+1} + f_{n+2})\ U_k^j \mid 1 \le j \le n\}$

$= (f_{n+1} + f_{n+2})\Sigma\{f_j\ U_k^j \mid 1 \le j \le n\}$

$= 0$ puisque U_k est centrée.

b) $\langle U_{n+1}, U_k \rangle = 0$ car $\forall\, j \in \{1, \ldots, n+2\}$ U^j_{n+1} ou U^j_k est nul.

Il reste à montrer que $\langle U_n, U_{n+1} \rangle = 0$:

$$\langle U_n, U_{n+1} \rangle = \Sigma\{f_j\, U^j_n\, U^j_{n+1} \mid 1 \le j \le n+2\}$$
$$= -f_{n+1}(\Sigma\{f_j \mid 1 \le j \le n\}) f_{n+2} + f_{n+2}(\Sigma\{f_j \mid 1 \le j \le n\}) f_{n+1} = 0$$

B - Le nombre n de composantes est impair (≥ 3)

Construisons par la méthode précédente la matrice du système orthogonal d'ordre impair formé de n-2 fonctions centrées à n-1 composantes et bordons cette matrice par la ligne et la colonne suivantes :

	$U_1 \cdots U_{n-2}$	U_{n-1}
1 . . . n-1	Système orthogonal de fonctions centrées	f_n ⋮ f_n
n	0 0	$-\Sigma\{f_j \mid 1 \le j \le n-1\}$

Les colonnes $U_1, \ldots, U_{n-2}$ restent évidemment orthogonales et centrées.

Montrons que U_{n-1} est centrée :

$$\Sigma\{f_j\, U^j_{n-1} \mid 1 \le j \le n\} = \Sigma\{f_j\, f_n \mid 1 \le j \le n-1\} - f_n \Sigma\{f_j \mid 1 \le j \le n-1\}$$
$$= f_n \Sigma\{f_j \mid 1 \le j \le n-1\} - f_n \Sigma\{f_j \mid 1 \le j \le n-1\}$$
$$= 0$$

Montrons l'orthogonalité :

$\forall\, k \in \{1, \ldots, n-2\}$

$$\langle U_{n-1}, U_k \rangle = \Sigma\{f_j\, U^j_{n-1}\, U^j_k \mid 1 \le j \le n\}$$
$$= \Sigma\{f_j\, f_n\, U^j_k \mid 1 \le j \le n\}$$
$$= f_n\, \Sigma\{f_j\, U^j_k \mid 1 \le j \le n\}$$
$$= 0 \text{ puisque } U_k \text{ est centrée.}$$

Ceci achève quel que soit $n \ge 2$ la construction d'un système de fonctions orthogonales et centrées d'ordre n-1.

4.3 Normalisation

Plusieurs cas sont à envisager selon que k, qui varie entre 1 et n-1, est pair ou impair.

- k est impair. Alors $U^j_k = 0$ si $j \le k-1$ ou $j \ge k+2$

$= f_{k+1}$ si $j = k$

$= -f_k$ si $j = k+1$

$$\| U_k \|^2 = f_k (f_{k+1})^2 + f_{k+1} (f_k)^2 = f_k\, f_{k+1} (f_k + f_{k+1})$$

d'où la fonction normée U'_k :

$$U'^{j}_{k} = 0 \text{ si } j \leq k-1 \text{ ou } j \geq k+2$$
$$= \left(\frac{f_{k+1}}{f_k (f_k+f_{k+1})}\right)^{1/2} \text{ si } j = k$$
$$= -\left(\frac{f_k}{f_{k+1}(f_k+f_{k+1})}\right)^{1/2} \text{ si } j = k+1$$

- k est pair. Il faut distinguer deux cas :

a) $k \leq n-2$. Alors : $U^j_k = f_{k+1} + f_{k+2}$ si $1 \leq j \leq k$

$$= -\Sigma\{f_j \mid 1 \leq j \leq k\} \text{ si } k+1 \leq j \leq k+2$$
$$= 0 \text{ sinon}$$

$$\|U_k\|^2 = \Sigma\{f_j (f_{k+1} + f_{k+2})^2 \mid 1 \leq j \leq k\} + (f_{k+1} + f_{k+2})(\Sigma\{f_j \mid 1 \leq j \leq k\})^2$$
$$= (f_{k+1} + f_{k+2})(\Sigma\{f_j \mid 1 \leq j \leq k\})(\Sigma\{f_j \mid 1 \leq j \leq k+2\})$$

d'où la fonction normée U'_k :

$$U'^{j}_{k} = \left(\frac{f_{k+1} + f_{k+2}}{(\Sigma\{f_j \mid 1 \leq j \leq k\})(\Sigma\{f_j \mid 1 \leq j \leq k+2\})}\right)^{1/2} \text{ si } 1 \leq j \leq k$$

$$= -\left(\frac{\Sigma\{f_j \mid 1 \leq j \leq k\}}{(f_{k+1}+f_{k+2})(\Sigma\{f_j \mid 1 \leq j \leq k+2\})}\right)^{1/2} \text{ si } k+1 \leq j \leq k+2$$

$$= 0 \text{ sinon}$$

b) $k = n-1$. Alors : $U^j_{n-1} = f_n$ si $1 \leq j \leq n-1$

$$= -\Sigma\{f_j \mid 1 \leq j \leq n-1\} \quad \text{si } j = n$$

$$\|U_{n-1}\|^2 = \Sigma\{f_j (f_n)^2 \mid 1 \leq j \leq n-1\} + f_n(\Sigma\{f_j \mid 1 \leq j \leq n-1\})^2$$
$$= f_n(\Sigma\{f_j \mid 1 \leq j \leq n-1\})(\Sigma\{f_j \mid 1 \leq j \leq n\})$$

d'où la fonction normée U'_{n-1} :

$$= \left(\frac{f_n}{(\Sigma\{f_j \mid 1 \leq j \leq n-1\})(\Sigma\{f_j \mid 1 \leq j \leq n\})}\right)^{1/2} \text{ si } 1 \leq j \leq n-1$$

$$= -\left(\frac{\Sigma\{f_j \mid 1 \leq j \leq n-1\}}{f_n(\Sigma\{f_j \mid 1 \leq j \leq n\})}\right)^{1/2} \text{ si } j = n$$

Enfin pour obtenir une base orthonormée il suffit d'ajouter la fonction contante U_n : $\forall\ 1 \leq j \leq n \quad U^j_n = (\Sigma\{f_j \mid 1 \leq j \leq n\})^{-1/2}$

Remarque 1 .

Le système peut être conservé dans un tableau à deux lignes et n-1 colonnes. Il suffit en effet de ne garder pour $k \in \{1, \ldots, n-1\}$ que les composantes U^k_k et U^{k+1}_k de U_k.

Remarque 2 .

Dans l'analyse des tableaux logiques mis sous forme disjonctive complète (cf[BIN.MULT.], Cahiers, Vol II n° 1, pp 55-71) , Q désigne l'ensemble des questions, J_q l'ensemble des modalités de réponses à une question $q \in Q$, $J = \Sigma\{J_q \mid q \in Q\}$ la somme ensembliste (i.e. ensemble des (j,q) avec $j \in J_q$ de toutes les modalités de toutes les questions), I l'ensemble des individus enquêtés et k_{IJ} le tableau disjonctif complet (k_{ij} = 1 si i choisit la modalité j, 0 sinon). On a :

$$k_J = \{k_j = \Sigma\{k(i,j) \mid i \in I\} \mid j \in J\}$$

$$k = \Sigma\{k_j \mid j \in J\} = \text{Card } Q \quad \text{Card } I$$

$$f_{IJ} = k_{IJ}/k \; ; \; f_J = k_J/k \text{ etc.}$$

On rappelle que les Card Q fonctions constantes δ^{Jq} valant 1 sur leur support J_q sont des facteurs triviaux et que les facteurs qui nous intéressent sont orthogonaux aux δ^{Jq}, i.e. sont dans H^J, ensemble des fonctions sur J de moyenne nulle sur chaque J_q. Ces facteurs sont obtenus par diagonalisation de $S = \frac{1}{k \text{ Card } Q} \; {}^t U B U$ où U est une matrice dont les colonnes forment une base orthonormée de H^J et B le tableau de Burt associé au questionnaire :

$$(\forall\, j, j' \in J \quad B_{jj'} = \Sigma\{k(i,j)\; k(i,j') \mid i \in I\}).$$

On choisit dans H^J une base orthonormée composée de Card Q blocs. Chaque bloc q est formé de Card J_q -1 fonctions de support J_q centrées et orthonormées dont la construction est décrite ci-dessus. On diagonalise ainsi une matrice d'ordre Card J - Card Q.

Dans notre chaîne de traitement nous utilisons la pondération entière k_J au lieu de f_J car on a : $S = \frac{1}{\text{Card } Q} \; \frac{{}^t U}{\sqrt{k}} \; B \; \frac{U}{\sqrt{k}}$; les colonnes $U_\ell/\sqrt{k}$ étant orthonormées(*), et calculées à partir des k_j.

Remarque 3.

$\forall\, 1 \le k \le n$ les composantes des U_k sont de la forme générale :

$\left(\frac{a_k}{b_k(a_k + b_k)}\right)^{1/2}$ ou $-\left(\frac{b_k}{a_k(a_k + b_k)}\right)^{1/2}$. On peut donc utiliser les vecteurs $A = \{a_k \mid 1 \le k \le n\}$ $B = \{b_k \mid 1 \le k \le n\}$ que si leurs composantes sont entières, nécessitent un encombrement moindre en mémoire centrale.Cependant pour le calcul de ${}^t U B U$ par exemple il faut calculer à chaque utilisation les U_k^k et U_k^{k+1}, solution qui accroît considérablement les temps de calcul et que nous n'avons pas retenue dans nos programmes.

4.4 *Exemples de programmes commentés*

Voir listings.

(*) pour k_J.

```
0: "CALCUL D'UNE BASE FJ-ORTHONORMEE : U[2,N]":
1:
2: "Le symbole ) represente l'assignation: 1)A signifie A=1 en FORTRAN":
3: "Le symbole \ represente la fonction SQRT en FORTRAN":
4: "Le dernier vecteur est constant":
5:
6: "N doit etre superieur ou egal a 2":
7: if N<2;gto "end"
8:
9: "Aucun Fj ne doit etre nul !":
10: for J=1 to N
11: if F[J]=0;gto "end"
12: next J
13:
14: 0)G
15: 0)U[1,N-1]
16: for K=1 to N-1 by 2
17: F[K]+F[K+1])F
18: \(F[K+1]/(F[K]*F)))U[1,K]
19: -\(F[K]/(F[K+1]*F)))U[2,K]
20: F+G)H
21: if K=1;gto +3
22: \(F/(G*H)))U[1,K-1]
23: -\(G/(F*H)))U[2,K-1]
24: H)G
25: next K
26: if U[1,N-1]#0;gto +4
27: F[N]+G)H
28: \(F[N]/(G*H)))U[1,N-1]
29: -\(G/(H*F[N])))U[2,N-1]
30: 1/\H)U[1,N]
31: 1/\H)U[2,N]
32: "end":end
```

- Calcul de B U :

Soit U' la matrice d'un système orthonormé de n-1 vecteurs centrés à n composantes, U sa forme compacte (2 lignes) et B une matrice (p,n) , on a :

$$(B\,U')_{jl} = \Sigma\{B_{jk}\ U'_{kl} / 1 \le k \le n\}$$

U'_{kl} étant la $k^{ième}$ composante du $l^{ième}$ vecteur de la base considérée ; l étant fixé, soit r_3 le plus petit indice k tel que U'_{kl} soit $\neq 0$ ($r_3 = 1$ ou l) et r_4 le plus grand indice k tel que U'_{kl} soit $\neq 0$ ($r_4 = l+1$ ou $l+2$) on a :

$$(B\,U')_{jl} = \Sigma\{B_{jk}\ U'_{kl} \ / \ r_3 \le k \le l\} + \Sigma\{B_{jk}\ U'_{kl} \ / \ l+1 \le k \le r_4\}$$

$$= (\Sigma\{B_{jk} / r_3 \le k \le l\})\,U_{1l} + (\Sigma\{B_{jk} / l+1 \le k \le r_4\})\,U_{2l}$$

$$= R\,U_{1l} + S\,U_{2l}$$

```
0: "Voici deux exemples de calcul du produit B[N,N]*U[2,N-1]=C[N,N-1]":
1:
2: "Le symbole ) represente l'assignation: 1)A signifie A=1 en FORTRAN":
3: "Les  variables r3 et r4 sont entieres":
4: "****************************************":
5: "Premier exemple : boucle sur L croissante":
6: 2)r4;for L=1 to N-1
7: if L#r4;L)r3;gto +3
8: 1)r3
9: min(r4+2,N))r4
10: for J=1 to N
11: 0)R;for K=r3 to L
12: B[J,K]+R)R;next K
13: 0)S;for K=L+1 to r4
14: B[J,K]+S)S;next K
15: R*U[1,L]+S*U[2,L])C[J,L];next J;next L
16: "****************************************":
17: "Deuxieme exemple : boucle sur L decroissante":
18: 1+Nmod2)r3
19: N)r4;for L=N-1 to 1 by -1
20: if r3#1;1)r3;gto +3
21: L)r3
22: L+1)r4
23: for J=1 to N
24: 0)R;for K=r3 to L
25: B[J,K]+R)R;next K
26: 0)S;for K=L+1 to r4
27: B[J,K]+S)S;next K
28: R*U[1,L]+S*U[2,L])C[J,L];next J;next L
```

```
0: "Exemple de calcul du produit U[2,N-1]*P[N-1,P]=Q[N-1,P]":
1:
2: "Le symbole ) represente l'assignation: 1)A signifie A=1 en FORTRAN":
3: "Le produit s'effectue en considerant que les colonnes de P":
4: "sont des combinaisons lineaires des colonnes de U":
5:
6: 2)M
7: for J=1 to N-1
8: U[1,J])A
9: U[2,J])B
10: if J#M;J)L;gto +3
11: 1)L
12: min(M+2,N))M
13: for K=1 to P
14: A*P[J,K])V
15: for I=L to J
16: V+Q[I,K])Q[I,K];next I
17: B*P[J,K])V
18: for I=J+1 to M
19: V+Q[I,K])Q[I,K];next I
20: next K
21: next J
```

Soit U et V les formes compactes des matrices U' et V' . r_3 et r_4 étant définis comme précédemment, fixons i et soir r_1 le plus petit indice j tel que V'_{ji} soit $\neq 0$ (r_1 = 1 ou i) et r_2 le plus grand indice j tel que V'_{ji} soit $\neq$ 0 (r_2 = l + 1 ou i + 2)

$$({}^{t}V' \; BU')_{i,l} = \Sigma\{V'_{ji} \; \Sigma\{B_{jk} \; U'_{kl} / r_3 \leq k \leq r_4\} / r_1 \leq j \leq r_2\}$$

$$= V_{1i}(\Sigma\{B_{jk} \; U'_{kl} | r_3 \leq k \leq r_4 \; , \; r_1 \leq j \leq i\}) + (V_{2i}(\Sigma\{B_{jk} U'_{kl} | r_3 \leq k \leq r_4 \; , i+1 \leq j \leq r_2\})$$

$$= V_{1i}(AU_{1l} + BU_{2l}) + V_{2i}(CU_{1l} + DU_{2l})$$

avec

$$A = \Sigma\{B_{jk} | r_3 \leq k \leq l \quad , \; r_1 \leq j \leq i\}$$

$$B = \Sigma\{B_{jk} | l + 1 \leq k \leq r_4 \; , \; r_1 \leq j \leq i\}$$

$$C = \Sigma\{B_{jk} | r_3 \leq k \leq l \quad , \; i + 1 \leq j \leq r_2\}$$

$$D = \Sigma\{B_{jk} | l + 1 \leq k \leq r_4 \; , \; i + 1 \leq j \leq r_2\}$$

Le listing du programme correspondant est donné ci-après :

```
0: "(Transpose de V[2,M-1]) * ( B[N,N] * U[2,N-1] ) = S[M-1,N-1] ":
1:
2: "Le symbole ) represente l'assignation: 1)A signifie A=1 en FORTRAN":
3: "Les variables r1,r2,r3,r4 sont entieres":
4: "Les boucles I et L peuvent etre independemment croissantes ou non":
5:
6: 2)r2
7: for I=1 to M-1
8: if I#r2;I)r1;gto +4
9: 1)r1
10: min(r2+2,M))r2
11:
12: 2)r4
13: for L=1 to N-1
14: if L#r4;L)r3;gto +4
15: 1)r3
16: min(r4+2,N))r4
17:
18: 0)A)B)C)D
19: for J=r1 to I
20: for K=r3 to L
21: A+B[J,K])A;next K
22: for K=L+1 to r4
23: B+B[J,K])B;next K
24: next J
25:
26: for J=I+1 to r2
27: for K=r3 to L
28: C+B[J,K])C;next K
29: for K=L+1 to r4
30: D+B[J,K])D;next K
31: next J
32:
33: V[1,I]*(A*U[1,L]+B*U[2,L])+V[2,I]*(C*U[1,L]+D*U[2,L]))S[I,L]
34: next L;next I
```

Annexe

Les programmes suivants commentés au § 3 et illustrant la compacité du langage sont extraits de la thèse soutenue par l'auteur en Octobre 1978.

```
0: "n":on err "Z";ret "BUR02 .8/3/78"
1: "":ret (rdb(0))F)=18 or F=146
2: "P$,U$ ne peuvent pointer que 255 colonnes ":
3: "T$ ne peut pointer que 100 colonnes dans la matrice de BURT incomplete":
4: "Z":if bit(7,rds(1));dsp "oter la protection de la bande!";gto +0
5: dsp "Mettre le papier en position TOF";trk 1;fdf ;stp
6: dsp ;wtb 701,27,"E";71)Q;dim S,T,E
7: dim N[Q],M[Q],T[500],U[0:99],Y[500],A$[227+16],B$[227+16],C$[4],D$[68]
8: dim R$[2,Q],Y$[4Q],Z$[1318];buf "A",A$,1;buf "B",B$,1;buf "Z",Z$,1
9:
10: ent "Nom de l'etude?( 64 lettres max)",Z$;jmp Z$$"" and (len(Z$))N)<65
11: wtb 701,27,"n",27,"&k1S";" "}D$[1,int(.5(68-N))}M]
12: "+"}D$[M-1];for I=M to M+N+1;"-"}D$[I];next I;"+"}D$[I]
13: fxd 0;fmt 1,c,/,4c,/,c,2/;wrt 701.1,D$,D$[1,M-2],"! ",Z$[1,N]," !",D$
14:
15: fmt 1,"Nom de la question:",f2,"? (arret=0)"
16: fmt 2,/,"Q.",c,":Rep.0)",f,c,"Groupement? (0/1)";wtb 701,27,"n",27,"&k0S"
17: fmt 3,"Numero attribue a l'item:",f,"  ?"
18: fmt 4,"Il reste ",f," colonnes dans BURT",c
19: fmt 5,z,"Question ",c,b,": r(ponses de 0 ",2b,x,f,".",c
20: fmt 6,z,f4
21: fmt 7,"Nb d'items ",2c,"regroupement:",f3,/
22: fmt 8,"Question ",c,b," annul(e",b,2/;trk 1;ldf 0,Z$
23:
24: 1}T;0}I}S
25: wrt .1,I+1;""}C$;ent "",C$[1}N,4];if C$="0   ";gto +25
26: jmp ((N+(pos(Z$[N],C$))}J)}N)mod7=2)-(not J)
27: if (num(Z$[N+3]))M))2;wrt .2,C$,M-1,".";(rdb(0))F)=48 or F=78 or F=206}C
28: dsp ;inq U;-(1}U[0])}K;for J=0 to M-1}D;if C;J}U[J]}rJ;gto +8
29: wrt .3,J;ent "",E;if flg13 or int(J)#J;beep;gto +0
30: if E}D or E<0;beep;dsp "L'item est hors des limites!";jmp -1-4''
31: if U[J]=J;gto +4;if E#J;beep;dsp "L'item",J," a ete retenu!";jmp -2-4''
32: if E}J;E}U[E];gto +2
33: if E<J;if U[E]#E;beep;dsp "L'item",E," n'existe plus!";jmp -4-4''
34: E}U[J];if E#J;gto +2
35: K+1}K}rJ
36: next J;if C;M}X;D}E;gto +3
37: (K}E)+1}X
38: if not K;beep;dsp "Il ne reste plus qu'un item!";jmp -11-2''
39: if (S-I}A)+E}Q;beep;wrt .4,Q-A,"!!";rdb(0))F;gto -14
40: wrt 701.5,C$,14,64,15,D
41: if C;wrt 701,"  Aucun groupement.";wrt 701;gto +2
42: wrt 701,"  Fonction de groupement d'items:"
43: for J=0 to D by 25;buf "A";buf "B";wtb "A","  item =";wtb "B","groupe ="
44: for L=J to min(J+24,D);wrt "A.6",L;wrt "B.6",U[L]}T[T+L}F]}U;rU}Y[F]
45: next L;if not C;wrt 701,A$[1,rds("A")}C];wrt 701,B$[1,C];wrt 701
46: next J;dsp "Correct?(oui=continue)"
47: if (rdb(0))F)#25 and F#153;wrt 701.8,C$,14,15;gto -22
48: wrt 701;C$}Y$[4I+1];Z$[N+4]}R$[1,I+1}I,I];Z$[N+5]}R$[2,I,I]
49: S+(X}M[I])}S;T+(M}N[I])}T;wrt .4,Q-S+I;rdb(0))F;if S-I<Q;gto -24
50: if not I;beep;dsp "Aucune question n'a ete retenue!";rdb(0))F;gto -25
51:
52: I}Q;if S=(T-1}T);wrt 701.7,":"&str(T);wrt 701;gto +2
53: wrt 701.7,"avant"," ",T;wtb 701,14;wrt 701.7,"apr}s"," ",S;wtb 701,15
54: 0}E;ent "Effectif analyse?(tout=continue)",E
55: if E;wtb 701,14,"Effectif analys( : ",15,str(E);wrt 701;gto +2
56: wrt 701,"Analyse de tous les individus"
57: wtb 701,12
58: dim K$[S,6}W],M$[Q],N$[Q],O$[T],P$[Q],U$[Q],Q$[4Q],T$[T]
59: fmt z,c,f2;Y$[1,4Q]}Q$;R$[1,1,Q]}P$;R$[2,1,Q]}U$
60: conv 32,95;0}L;for J=1}G}H to Q;char(N[J])}N$[J];char(M[J]}M)}M$[J]
61: (4J}V)-3}U;for I=0 to N[J]-1;char(Y[H+I}K])}Y)}O$[K]
62: if I=T[H+I];" "}Z$[1,W];buf "Z";wrt "Z",Q$[U,V],I;Z$[1,W]}K$[L+1}L]
63: Y+G-J+1}B;if M#Y+1;char(B)}T$[K];gto +2
64: char(100+B)}T$[K]
65: next I;N[J]+H}H;G+M}G;next J
66:
67: trk 1;rcf 1,Q,S,T,E;rcf 2,Q$;rcf 3,N$;rcf 4,T$;rcf 5,P$;rcf 6,U$
68: trk 1;rcf 7,M$;rcf 8,O$;rcf 9,K$;trk 0;ldp 6
```

```
0: "n":on err "Z";ret "BUR09. tU*B*U. 8/3/78"
1: "Z":dsp "Calcul de tU*B*U"
2: idf r0;trk 1;ldf 1,Q,I,N,E,S,U,A;S-U)N
3: dim M$[Q],K$[2I],U$[4N],V$[4N],B$[I-Q)M,4M]
4: ldf 12,B$;ldf 13,M$;ldf 14,K$;ldf 15,U$;ldf 16,V$
5: num(M$[1])-1)B;for J=2 to U;(B)G)+1)A;(B+(num(M$[J]))C)-1)B)-1)r2
6: bit(0,C))Q;A)X;for Z=J-1 to 1 by -1;(X)N)-1)Y;X-num(M$[Z])+1)X;0)D
7: for L=G to r2;fts (D+itf(K$[2(L+J)-1]))D))B$[N,4L-3,4L];next L
8: fts (itf(K$[2(N+Z)-3]))B$[N,4B-3,4B]
9: for I=X to Y;0)D;4G)E;for L=A to B;fts (stf(B$[I,4L-3])+D)D))B$[I,E-3,E]
10: fts (stf(B$[N,E-3])-D))B$[N,E-3,E];E+4)E;next L
11: fts (itf(K$[2(I+Z)-3]))B$[I,4B-3,4B];next I;X+1)H
12: for I=X to Y;stf(U$[4I-3]))R;stf(V$[4I-3]))S
13: I+1)U;if I=H;X)M;min(H+2,N))H;gto +2
14: I)M
15: Q)P;for L=B)F to A by -1;4(L-1)T)+1)r1;if P;0)P;gto +2
16: (L)F)-2)P
17: 0)O)C)D)E)V)W;for K=M to I;if P)=A;stf(B$[K,4P-3])+O)O
18: stf(B$[K,4T-3])+C)C
19: stf(B$[K,4F-3])+D)D;next K;for K=U to H;if P)=A;stf(B$[K,4P-3])+E)E
20: stf(B$[K,4T-3])+V)V;stf(B$[K,4F-3])+W)W;next K
21: fts (stf(U$[r1])(R(C-O)+S(V-E))+stf(V$[r1])(R(D-C)+S(W-V))))B$[L,4I-3,4I]
22: next L;next I;next Z;next J
23: for J=1 to B;fts (1))B$[J,4J-3,4J];next J
24: trk 1;rcf 17,B$
25: dsp ;trk 0;ldp r0
```

```
0: "n":on err "Z";ret "BUR06.14/8/78. Edition du tableau de BURT (complet)"
1: "Z":wtb 701,27,"&k0SBURT ",14,"(tris crois(s)",15,27,"&k2S",13,10,10
2: fmt 1,z,f5.0;0)M)O;1)D)P
3: M)Y;for I=P to Q;Y+D)X;Y+num(P$[I]))Y;if Y+I-O)45.4;gto +2;if I=P;end
4: I)T
5: for L=X to Y;4(L-I)+D)V;2L-D)E
6: M)B;for N=P to T;B+D)C;B+num(P$[N]))B
7: 0)R;for K=C to B-D;if K=L;wrt 701.1,itf(K$[E]);gto +5
8: (4(K-N+D)F))H)-3)G;if L=X;fts (itf(K$[2K-D]))B$[F,G,H];gto +2
9: if L=Y;stf(B$[F,G]))Z;gto +2
10: fts (stf(B$[F,G])-(stf(B$[F,V]))Z))B$[F,G,H]
11: R+Z)R;wrt 701.1,Z;next K;wrt 701.1,itf(K$[E])-R,"     ";next N
12: wrt 701;next L;wtb 701,27,"&13V";next I;if B$Y;(B)M)+(T+1)P))O;gto -9
13: wtb 701,12;trk 0;ldp A+3+J
```