LES CAHIERS DE L'ANALYSE DES DONNÉES

ALEXANDROS KARAKOS

Un algorithme performant pour le calcul des valeurs propres et des vecteurs propres en analyse factorielle des correspondances

Les cahiers de l'analyse des données, tome 13, n° 2 (1988), p. 205-222

<http://www.numdam.org/item?id=CAD_1988__13_2_205_0>

© Les cahiers de l'analyse des données, Dunod, 1988, tous droits réservés.

L'accès aux archives de la revue « Les cahiers de l'analyse des données » implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM

Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Les Cahiers de l'Analyse des Données
Vol XIII 1988 - n°2 - pp. 205-222

UN ALGORITHME PERFORMANT POUR LE CALCUL DES VALEURS PROPRES ET DES VECTEURS PROPRES EN ANALYSE FACTORIELLE DES CORRESPONDANCES

[ALG. V. V. PROPRES]

Alexandros KARAKOS (*)

1. Introduction

En général il y a deux manières de calculer les valeurs propres et les vecteurs propres. Ou bien on calcule une valeur propre et le vecteur correspondant (méthode de Golub et Reinsch [2]), ou bien on calcule toutes les valeurs propres et ensuite les vecteurs (méthode de Givens-Householder [3]). On sait qu'une fois connue la valeur propre λ_α, le vecteur propre correspondant V_α se calcule facilement.

En analyse factorielle des correspondances (a.f.c.), on travaille, ordinairement, sur les 4-5 premiers facteurs: il suffit donc d'obtenir les 4-5 premières valeurs propres, de calculer les vecteurs propres correspondants, de normaliser ces vecteurs et ensuite de calculer les facteurs. Il est inutile de calculer toutes les valeurs propres afin d'avoir le taux d'inertie afférent à chaque axe, car la somme des valeurs propres, ou trace, se calcule directement.

Une autre remarque est qu'en a.f.c. on travaille sur un tableau carré symétrique. On garde donc en mémoire deux fois les mêmes valeurs ou plutôt on travaille sur la partie inférieure du tableau sans rien faire sur la partie supérieure. Or ces places mémoire inutilisées sont quelquefois si importantes, notamment sur les ordinateurs de taille moyenne, que les utilisateurs sont incapables de traiter leurs données. Le sous-programme VVPROM (Valeurs, Vecteurs, PROpres, version Mono-indicé) corrige dans la mesure du possible

(*) Professeur à l'Université DEMOKRITOS de Thrace; Grèce.

Les cahiers de l'analyse des données – 0339-3097/88/02 205 18/$ 3.80/ © Gauthier-Villars

tous ces inconvénients en économisant d'une part l'espace mémoire et de l'autre le temps total de calcul.

2. Principes de l'algorithme

2.1. Comment économiser l'espace mémoire

En a.f.c. le tableau sur lequel on travaille pour trouver les valeurs propres et les vecteurs propres est un tableau carré symétrique. Ainsi on garde la diagonale et les élénents au-dessous de la diagonale et on ne fait de calculs que sur cette partie inférieure du tableau. Tous les programmes qui existent pour les calculs de valeurs propres et de vecteurs propres réservent aussi en mémoire la place de la partie supérieure du tableau. Il est vrai que dans ce même tableau, à la fin du calcul, on garde la totalité des vecteurs propres; mais puisque, en a.f.c. le nombre des facteurs demandés par l'utilisateur dépasse très rarement dix, nous pouvons économiser la mémoire en gardant d'une part un tableau où on aura seulement les éléments qui sont au-dessous de la diagonale; et d'autre part un deuxième tableau où seront gardés les vecteurs propres nécessaires pour calculer les facteurs demandés.

Les éléments d'un tableau carré N×N qui sont au-dessous de la diagonale avec les élénents de la diagonale sont au total N*(N+1)/2: je propose, ici, au lieu d'un tableau carré N×N , d'utiliser un tableau à une dimension (mono-indicé) de longueur N*(N+1)/2. La correspondance entre les éléments du tableau carré et les éléments de ce tableau mono-indicé se fait très facilement à l'aide de la formule classique du numérotage des paires dans l'ordre lexicographique :

$$\text{rang}(U,V) = \inf(U,V) + ((\sup(U,V)-1)*\sup(U,V)/2)$$

Mais cette formule nécessite pour chaque élément les quatre opérations de base (addition, soustraction, multiplication et division). Nous pouvons donc, afin de réduire les opérations, utiliser un tableau auxiliaire de dimension N. Dans ce tableau on met les valeurs calculées par la suites des instructions (1) et qui réprésentent le nombre des éléments qui se trouvent au-dessous et sur la diagonale avant chaque ligne du tableau carré initial.Ainsi on écrit les trois instructions (1) en FORTRAN pour remplir ce tableau; soit IT de dimension N, qu'on appelle tableau des indices :

```
      IT(1)=0
      DO  1  I=2,N                                    (1)
1     IT(I)=IT(I-1)+I-1
```

Pour mettre maintenant les éléments du tableau carré, soit T de dimension N×N, dans le tableau mono-indicé, soit F de dimension (N*(N+1))/2, il suffit

d'écrire les quatre instructions (2) en FORTRAN. Chaque fois qu'on a besoin d'un élément du tableau carré symétrique T,il suffit de trouver l'indice II, par une addition, qui donne directement la position de cet élément dans le tableau F. Si on cherche, par exemple, l'élément t32 du tableau d'origine T, on trouve l'indice II = IT(3)+2 = 5 et on recupère le 5-ème élément du tableau F.

Les instructions pour remplir le tableau F sont :

```
      DO 2 I=1,N
      DO 2 J=1,I
      II = IT(I)+J                                    (2)
2     F(II)=T(I,J)
```

Un deuxième tableau est nécessaire pour garder les vecteurs propres: c'est un tableau à deux dimensions; l'une est la longueur des composantes du vecteur et l'autre le nombre de facteurs demandé par l'utilisateur.

Ainsi on occupe le minimum possible de l'espace mémoire en fonction des paramètres d'analyse.

2.2. Les étapes de l'algorithme

L'algorithme proposé est divisé en trois étapes. Au début on réduit le tableau initial à une forme tridiagonale et ensuite on sépare la recherche des valeurs propres du calculs des vecteurs propres.On a donc les étapes suivantes :

a. Tridiagonalisation

Pour la tridiagonalisation de la matrice carrée symétrique N×N on utilise la méthode de Householder modifiée pour une matrice à une dimension de longueur (N*(N+1)/2 qui est la disposition linéaire des éléments du tableau initial exposée au §2.1. Les détails de la méthode de Householder se trouvent dans [3], [4] et [5].

b. Recherche des valeurs propres

Contrairement aux méthodes décrites dans [2] et [7], où après le calcul de chaque valeur propre on calcule le vecteur propre correspondant, on adopte ici une suggestion de Givens [8]. On calcule d'abord toutes les valeurs propres, puisque l'équation aux valeurs propres de la matrice tridiagonale revêt une forme simple, et que les racines se calculent rapidement.

c. Calcul des vecteurs propres

On calcule autant de vecteurs propres qu'il y a de facteurs demandés. La méthode adoptée est celle décrite en détail dans [3] et [9], mais modifiée pour

une matrice à une dimension et l'orthonormalisation de la suite des vecteurs calculés s'obtient par l'algorithme de Schmidt et Hilbert [1] (p.p. 273-276).

3. Présentation du sous-programme VVPROM

En entrée on trouve :

a. Le tableau A, à une dimension, qui contient les élémentsdu tableau carré initial
b. Le tableau des indices IT, à une dimension
c. La variable N, qui exprime le nombre des variables du tableau carré initial
d. La variable NVEC, qui est égale au nombre des vecteurs propres à calculer
e. La variable IQ, qui est une variable auxiliaire pour l'impression des résultats

En sortie on obtient :

a. Le tableau E, à une dimension, qui contient les valeurs propres calculées
b. Le tableau VEC, à deux dimensions, qui contient les vecteurs propres orthonormés
c. La variable TEST, qui est égale à la somme de valeurs propres, moins la première valeur propre triviale égale à 1.

Les tableaux B,C,P,Q,R,W,Y, chacun à une dimension de longueur N, sauf le tableau Y qui est de longueur N+2 , sont des tableaux auxiliaires pour les calculs intermédiaires. Les éléments de la diagonale sont dans le tableau C, et le tableau B contient les N-1 éléments de la sous-diagonale tandisque le dernier, le B(N), est mis à zéro. Les éléments de la sur-diagonale sont par symétrie égaux à ceux de la sous-diagonale.Les valeurs propres sont gardées dans le tableau E.

En a.f.c. les valeurs propres se trouvent entre 1 et 0 mais on ne cherche pas de valeurs plus petites que 1.0*10-5. En réalité dans tous les programmes de calcul des valeurs propres [11] ,on trouve quelques valeurs inférieures à ce seuil et quelques valeurs négatives du même ordre en valeur absolue, faute de précision dans les calculs. Ainsi, dans ce sous-programme il y a un seuil EPS=1.0*10-6 et on ne cherche pas de valeurs propres inférieures à EPS.

À la suite on calcule les NVEC vecteurs propres. Puisqu'on calcule en plus le premier vecteur trivial, NVEC prend une valeur égale à NF+1, où NF=nombre des facteurs demandés pour l'analyse.

Pour la résolution des systèmes obtenus on utilise la méthode de CROUT [10]. Les vecteurs calculés sont gardés dans le tableau VEC en colonnes et dans la même ordre que les valeurs propres. Les vecteurs orthonormés se trouvent rangés dans le même tableau VEC en colonnes.

4. Résultats de tests - Performances

Toutes les comparaisons faites dans ce paragraphe sont entre le nouveau sous-programme VVPROM et le sous-programme SYMQR comme il est présenté dans le programme BENTAB2 (1978) [1], et sont effectuées sur une machine UNIVAC 90/30.

4.1 Comparaisons de taille

La formule qui donne la taille de l'espace mémoire en mots machine qu'occupent les tableaux utilisés dans le sous-programme SYMQR est :

4*NJ+NJ*NJ ; où NJ = nombre des variables

La formule pour le sous-programme VVPROM est :

9*NJ+NJ*NF+(NJ*(NJ+1))/2

où NJ=nombre des variables ; NF=nombre des facteurs demandés.

Avec NF=6, on trouve que pour NJ=23 les deux sous-programmes ont les mêmes besoins de mémoire. Ce qui signifie que jusqu'à 22 variables à analyser c'est SYMQR qui domine par rapport à la taille; mais à partir de 24 variables VVPROM domine. Le tableau 4.a. donne une idée des relations entre les besoins en mémoire des deux sous-programmes.

Nbre de variables	20	50	100	200	300	500	1000
SYMQR	480	2700	10400	40800	91200	252000	1004000
VVPROM	510	2025	6550	23100	49650	135000	515500

Tableau 4.a. Comparaison des deux sous-programmes pour différentes valeurs du nombre des variables à analyser, si NF=6.

4.2. Comparaison du temps de calcul

Si on remplit un tableau carré N×N par une formule simple de telle façon que tous les éléments sont symétriques par rapport à la diagonale et qu'ensuite on les dispose linéairement dans un tableau à une dimension nous pouvons très façilement comparer le temps de calcul par les deux sous-programmes.

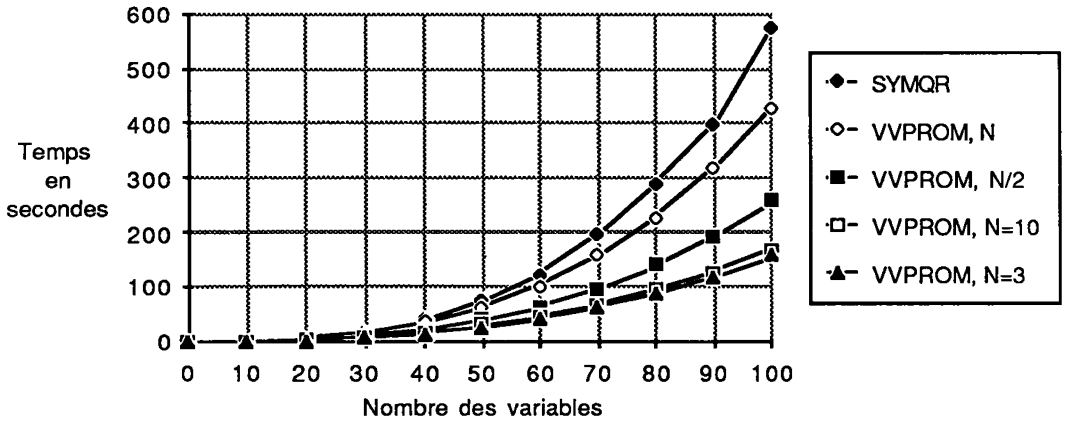
NB Le temps de calcul de la matrice à diagonaliser elle-même, temps qui peut être considérable si le nombre des individus est élevé, est compté ici seulement dans les tableaux 4.c et 4.d.

Dans le tableau 4.b. il y a les valeurs prises, en secondes, après l'exécution d'un programme de test pour N=10 jusqu'à N=100. On trouve aussi le temps de calcul quand le nombre des vecteurs propres calculés varie et prend les valeurs N, N/2, 10, 3. La figure 4.a. est la représentation graphique du contenu du tableau 4.b.

N	SYMQR	VVPRM(N)	VVPRM(N/2)	VVPRM(10)	VVPRM(3)
10	1,185	1,167	0,991	1,256	0,900
20	6,379	6,082	4,536	4,536	3,672
30	18,668	16,298	11,119	9,802	8,304
40	39,841	34,655	22,810	18,526	16,172
50	75,658	62,873	40,240	30,734	27,365
60	125,464	103,299	64,745	47,215	42,649
70	197,196	158,258	97,688	68,726	62,756
80	290,231	229,910	140,219	95,841	88,328
90	397,847	320,147	193,240	128,844	119,597
100	575,030	431,504	258,264	168,706	157,526

Tableau 4.b. Comparaison du temps calcul entre les sous-programmes SYMQR et VVPROM

Figure 4.a. Représentation graphique du tableau 4.b.



4.3. Comparaison de résultats

Soit A la suite des axes orthogonaux et α l'indice qui parcourt A. La somme de tous les moments d'inertie qui s'appelle aussi trace est :

$$T = \Sigma\{ \lambda_\alpha \mid \alpha \in A \}$$

c.a.d. égal à la somme de toutes les valeurs propres.

À chaque moment d'inertie λ_α, il correspond une part de l'inertie totale, communément notée T_α (taux)

$$T_\alpha = \lambda_\alpha / \Sigma\{ \lambda_\alpha \mid \alpha \in A \}$$

Dans le sous-programme VVPROM on ne calcule pas toutes les valeurs propres mais, ce qui est plus précis, on calcule directement la trace de la matrice en faisant la somme des éléments de la diagonale du tableau d'origine.

On croyait qu'à cause des nombreux produits scalaires et des soustractions que comporte la méthode de Schmidt et Hilbert on aurait une accumulation d'erreurs d'arrondi dans le calculs des vecteurs orthonormés; mais, indifféremment, que l'on orthonormalise certains vecteurs ou tous les vecteurs, on trouve les mêmes valeurs par les deux sous-programmes.

Ainsi à partir du moment où l'on trouve les mêmes valeurs pour les vecteurs orthonormés, tous les calculs qui suivent pour trouver le poids, la qualité, l'inertie, les coordonnées sur les axes, les contributions et les corrélations donnent exactement les mêmes résultats.

Il faut noter ici, que les valeurs propres calculées de l'exemple sont positives et très grandes sauf la dernière qui est négative. En a.f.c. chaque valeur propre est nécessairement comprise entre 0 et 1 (cf. § 2.4 de [1]. On a donc essayé de tester les deux sous-programmes pendant l'exécution d'un programme complet d'a.f.c. avec NF=7. Le tableau 4.c. donne, dans ses deux premières colonnes, le temps de calcul, en sec., quand le nombre des variables varie de 10 à 100; puis, dans les deux colonnes suivantes, le temps total d'éxécution d'un programme d' a.f.c, si on utilise, pour un même jeu de données, (constituant un tableau carré), d'abord les sous-programme SYMQR et ensuite le sous-programme VVPROM. La dernière colonne du tableau 4.c. donne, en secondes, le gain réalisé, si on remplace le sous-programme SYMQR par VVPROM dans le programme d' a.f.c. (Voir §5 du présent article). Enfin, le tableau 4.d. concerne des tableaux 1000×100 et 100×50; avec un nombre aussi élevé d'individus (lignes) le temps requis pour créer la matrice à diagonaliser devient prépondérant.

Remarques

a. D'après le tableau 4.c., pour 40 variables, le temps de calcul pour le sous-programme VVPROM est la moitié du temps nécessaire pour SYMQR et si on va plus loin, aux environs de 120 variables, on arrive à une relation 1 : 3.

N	SYMQR	VVPROM	Pr(SYMQR)	Pr(VVPROM)	GAIN
10	1,183	0,846	56,136	55,446	0,690
20	4,316	3,109	63,708	62,102	1,606
30	13,146	7,518	76,085	70,136	5,949
40	29,242	14,449	96,906	82,518	14,388
50	55,102	24,819	129,216	100,113	29,103
60	91,809	39,202	175,616	125,619	49,997
70	148,517	58,192	246,227	160,912	85,315
80	212,718	81,506	372,416	212,353	160,063
90	299,105	111,846	448,441	260,207	188,234
100	403,902	148,231	574,105	334,748	239,357

Tableau 4.c. Temps de diagonalisation, avec SYMQR et VVPROM; temps total de calcul, avec ces deux sous-programmes; et gain.

b. D'après le tableau 4.c. si on remplace le sous-programme SYMQR par VVPROM dans un programme d' a.f.c. on a un gain toujours positif. Pour 100 variables, par exemple, on gagne 41.7% du temps total d'éxecution.

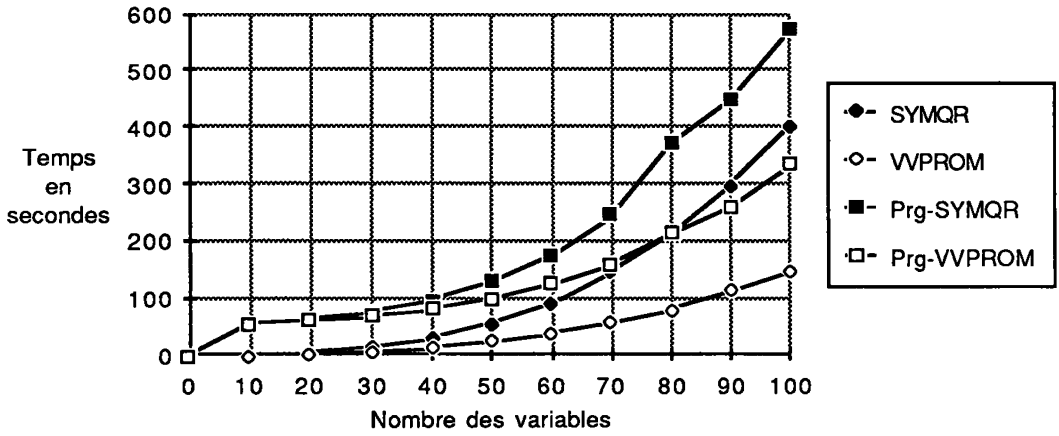


Figure 4b : Représentation graphique du contenu du tableau 4c

	100 variables	50 variables
lecture de 1000 individus	123,680 sc	82,349 sc
calcul de trois facteurs	8,289 sc	1,769 sc
calcul de tous les facteurs	23,883 sc	4,199 sc

Tableau 4.d. Temps de calcul pour une analyse complète

5. Utilisation du sous-programme VVPROM

5.1. Utilisation générale

Le sous-programme VVPROM, donné en annexe, peut être très facilement utilisé dans tout programme où on calcule les valeurs propres et les vecteurs propres. Pour utiliser deVVPROM, il faut prévoir dans le programme appelant :

a. Un tableau A de dimension (N*(N+1))/2 , où N=nombre des variables d'analyse

b. Un tableau IT de dimension N, qui est le tableau des indices et qu'il faut remplir dans le programme appelant suivant les instructions FORTRAN (1) du § 2.1.

c. Un tableau E de dimension N où seront gardées les valeurs propres dans l'ordre décroissant

d. Un tableau VEC à deux dimensions, N×NVEC où seront stockés les vecteurs propres orthonormés en colonnes et dans le même ordre décroissant que les valeurs propres. (NVEC=nombre des vecteurs demandés)

e. Six tableaux auxiliaires, chacun de dimension N (B,C,P,Q,R,W) et un tableau auxiliaire Y de dimension N+2.

La variable NEV exprime le nombre des valeurs propres calculées par le programme.

La variable TEST est la somme de toutes les valeurs propres calculées, moins la première valeur triviale pour l' a.f.c..

La variable IQ est une variable d'entrée

IQ = 1 signifie impression des vecteurs propres et des valeurs propres ainsi que leur histogramme

IQ = 0 signifie impression seulement des valeurs propres avec leur histogramme

Si on ne veut pas arrêter le calcul des valeurs propres au seuil EPS=1.0*10-6, mais les calculer toutes, il suffit de mettre EPS=-1.0*10-6

Le remplissage du tableau à un indice A, par les éléments du tableau carré symétrique de travail TAB, se fait par les instructions FORTRAN (2) du 2.1.

Ainsi après le calcul d'un élément, au lieu d' écrire :

```
TAB( I, J ) = . . . . . .
```

il faut mettre les deux lignes suivantes :

```
II = IT( I ) + J
A( II ) = . . . . . .
```

Remarque importante

On suppose que l'on travaille seulement dans la partie inférieure du tableau carré symétrique TAB. Si les calculs se font dans la partie supérieure on échange les rôles des indices I, J c.a.d. qu'au lieu d'écrire:

```
II=IT( I )+ J
```

on écrit :

```
II=IT( J )+ I
```

6. Utilisation dans le programme d' a.f.c. BENTAB2-1978-

Le sous-programme VVPROM peut remplacer le sous-programme SYMQR dans le programme BENTAB2(1978) [1], après les modifications suivantes :

a. Dans le sous-programme CORRES (pages 554-556)

On remplace les 10 dernières lignes de la page 555 par la suite des instructions :

```
NIJ=MAX(NI,NJ)
K=(NJ1*NJ1+NJ1)/2
I1=7*NJ+K
I2=2*NJ+K+NJ1*(NF+10)+2
I3=55*NF+10*NIJ+3*NIJ*NF
MEMOIR=MAX(I1,I2,I3)+1
PRINT 66, MEMOIR
```

```
66      FORMAT(1H0,18HPARAMETRE MEMOIR =,I10)
        CALL LECAC2(V(1),V(NJ+1),V(2*NJ+1),V(2*NJ+K+1),
     &  V(3*NJ+K+1),V(3*NJ+K+1),V(4*NJ+K+1),V(5*NJ+K+1),
     &  V(6*NJ+K+1),AKT,IOUT,NFOR,K)
        K1=2*NJ+K+NJ1
        K2=K1+NJ1*(NR+1)
        CALL VVPROM(V(2*NJ+1),V(2*NJ+1),V(K1+1),V(K1+NJ1+1),
     &  V(K2+1),V(K2+NJ1+1),V(K2+2*NJ1+1),V(K2+3*NJ1+1),
     &  V(K2+4*NJ1+1),V(K2+5*NJ1+1),V(K2+6*NJ1+1),NJ1,NR,
     &  TEST,NEV,IQ)
        IF(NR.LT.NF+1) NF=NR-1
        K3=K1+1
        K4=K+NJ1
        DO 55 I=K3,K2
55      V(I-K4)=V(I)
        K2=K2-K4
        NJF2=NJ2*NF
        IF(NJF2.EQ.0) NJF2=1
        NJD=K2+2*NJ+NJF2+1
        CALL FACACO(V(1),V(NJ+1),V(2*NJ+1),V(2*NJ+NJ1+1),
     &  V(K2+1),V(K2+NJ+1),V(K2+2*NJ+1),V(NJD),V(NJD+NF),
     &  V(NJD+2*NF),V(NJD+3*NF),V(NJD+4*NF),V(NJD+5*NF),
     &  V(NJD+6*NF),V(NJD+7*NF),V(NJD+8*NF),
```

b. Dans le sous-programme LECACO (page 556-558)

1. On remplace les 2 premières lignes du sous-programme par les instructions :

```
      SUBROUTINE  LECAC2(NOMJ,PJ,A,IT,NO,S1,IRANG,P,NOM,
     &     AKT,IOUT,NFOR,KIT)
      DIMENSION  FMT(60),TITRE(20),A(KIT),IT(NJ1),NOM(NJ),
     &     S1(NJ)
```

2. On remplace les trois instructions :

```
      DO  30  J= 1,NJ1
      DO  30  L= 1,NJ1
30    A(J,L)=0.0
```

par les instructions :

```
      IT(1)=0
      DO  30  J= 2,NJ1
30    IT(J)= IT(J-1)+J-1
      DO 29  J= 1,KIT
29    A(J)= 0.0
```

3. On remplace l'instruction :

```
33    A( J,JJ )=A( J,JJ ) + P( J )*P( JJ ) / PIIII
```

par les deux instructions :

```
      K=IT( J )+JJ
33    A( K )=A( K )+P( J )*P( JJ ) / PIIII
```

4. On remplace l'instruction :

```
533   A( J,JJ )=A( J,JJ )+P( J )*P( JJ ) / PIIII
```

par les deux instructions :

```
      K=IT( J )+JJ
533   A( K )=A( K )+P( J )*P( JJ ) / PIIII
```

5. On remplace l'instruction qui est avant l'étiquette 38 :

```
      A( J,JJ )=A( J,JJ) / SQRT( PJJJ*PJJJJ)
```

par les deux instructions :

```
K=IT( J )+JJ
A( K )=A( K ) / SQRT( PJJJ*PJJJJ)
```

c. On remplace la première ligne du sous-programme FACACO par l'instruction:

```
      SUBROUTINE FACACO( NOMJ,PJ,VAL,VEC,VARJ,P,FJS,
     &     FIM,KFIM,ICTR,INQ,
```

7. Conclusion

L'algorithme proposé est un algorithme très performant pour calculer les valeurs propres et les vecteurs propres d'une matrice carrée symétrique. Le sous-programme en FORTRAN, VVPROM , est une adaptation de l'algorithme pour le programme d' a.f.c. mais on peut l'utiliser dans tous les programmes où l'on a besoin de calculer les valeurs propres et les vecteurs propres d'une matrice carrée symétrique.

Si on compare VVPROM à SYMQR on constate que :

a. Quand le nombre des variables à analyser varie vers l'infini l'espace mémoire nécessaire pour les calculs est divisé par 2 dans VVPROM .

b. D'après le tableau 4.c. le temps de calcul pour VVPROM est toujours inférieur du temps nécessaire pour SYMQR.

Ainsi on peut remplacer facilement le sous-programme SYMQR par le sous-programme VVPROM dans tous les programmes d'analyse des données pour gagner du temps d'exécution et de l'espace mémoire.

8 Bibliographie

1. J.-P. Benzécri & coll., *L'analyse des données* , DUNOD, Tome II (1980) 3ème édition

2. G.H. Golub et C. Reinsch, Singular Value Decomposition and Least Squares Solution; *Num. Math.* 14, 403-420 (1970)

3. J. Ortega, *Mathematical Methods for digital computers*, Ralston & Wilf, Volume II (1968) Chap. 4 pp. 94-115.

4. A.S.Householder, *The theory of matrices in Numerical Analysis* Dover Publications (1975).

5. J.H. Wilkinson, Householder method for symetric Matrices, *Num. Math.* Vol. 4 pp 354-361 (1962)

6. J.H. Wilkinson, *The algebraic Eingenvalue problem*, Clarendon Press, Oxford (1965)

7. R.S. Martin, J.H. Wilkinson, Implicit QL Algorithm, *Num. Math.* Vol 12, pp. 377-383

8. J.W. Givens, A method of computing Eingenvalues and Eingenvectors suggested by classical results on symmetric matrices, *Nat. Bur. Stant. Appl Math. Ser.* 29 pp 117-122 (1953)

9. J.H. Wilkinson, The calculation of the Eingevector of codiagonal matrices, *Comp. Journal* Vol. 1, pp. 148-152 (1958)

10. J.P. Queri, A. Karakos, C. Bacoyannis, *Aide au choix d'une méthode de résolution matricielle* (Note technique EDF-GDF Ref. TC 79-(DF) JLQ-AK-CB).

11. Smith B.T., Boyle J.J., Bongarra J.J., GarboW B.S., Ikebe Y., Klema V.C, Moler C.B, *Matrix Eingensystem Routines-EISPACK guide* second edition; Springer-Verlag (1976).

9 Annexe: listage du sous-programme VVPROM

```
      SUBROUTINE VVPROM(A,IT,E,VEC,B,C,P,Q,R,W,Y,N,NVEC,TEST,NEV,IQ)
      DIMENSION A(1),B(1),C(1),P(1),Q(1),R(1),E(1),Y(1),W(1),IT(1)
      DIMENSION VEC(N,NVEC)
      DATA EPS,EEE,IAST/1.E-6,1.E-30,'*'/
      NM1=N-1
      NM2=N-2
      TEST=-1.0
      DO 33 I=1,N
      I1=IT(I)+I
33    TEST=TEST+A(I1)
C       ********************************************
C       *      REDUCTION A UNE FORME TRIDIAGONALE     *
C       ********************************************
      DO 8 I=1,NM2
      IP1=I+1
      SS=0.0
      DO 1 J=IP1,N
1     L1=IT(J)+I
       SS=SS+A(L1)*A(L1)
      S=SQRT(SS)
      L1=IT(IP1)+I
      IF(A(L1).LT.0.0) S=-S
      L11=IT(I)+I
      C(I)=A(L11)
      B(I)=-S
      ALPHA=0.0
      IF(S.EQ.0.0) GOTO 8
```

```
      ALPHA=1.0/(SS+A(L1)*S)
      T=A(L1)+S
      A(L1)=T
      W(I+1)=T
      IP2=I+2
      DO 2 J=IP2,N
      L2=IT(J)+I
2     W(J)=A(L2)
      DO 4 J=IP1,N
      T=0.0
      DO 3 K=IP1,N
      L2=IT(J)+K
      IF(J.LT.K) L2=IT(K)+J
3     T=T+A(L2)*W(K)
4     P(J)=T*ALPHA
      AP=0.0
      DO 5 K=IP1,N
5     AP=AP+W(K)*P(K)
      AP=0.5*AP*ALPHA
      DO 6 K=IP1,N
6     Q(K)=P(K)-AP*W(K)
      DO 7 J=IP1,N
      DO 7 K=J,N
      L2=IT(J)+K
      IF(J.LT.K) L2=IT(K)+J
7     A(L2)=A(L2)-(Q(J)*W(K)+Q(K)*W(J))
8     A(L11)=ALPHA
      L10=IT(N-1)+N-1
      C(N-1)=A(L10)
      L11=IT(N)+N
      C(N)=A(L11)
      L11=IT(N)+N-1
      B(N-1)=A(L11)
      B(N)=0.0
C  ************************************
C  *   RECHERCHE DES VALEURS PROPRES   *
C  ************************************
      RM=ABS(C(1))+ABS(B(1))
      DO 10 I=2,N
      T=ABS(C(I))+ABS(B(I))+ABS(B(I-1))
10    RM=AMAX1(RM,T)
      DO 11 I=1,NM1
11    W(I)=B(I)*B(I)
      K=0
      U=RM
      DO 12 I=1,N
12    E(I)=-RM
15    K=K+1
13    AL=E(K)
14    AMB=0.5*(AL+U)
      IF((AMB.EQ.AL).OR.(AMB.EQ.U)) GOTO 30
      IG=0
      I=1
16    S=C(I)-AMB
18    IF(S.GE.0.0) IG=IG+1
      IF(S.EQ.0.0) GOTO 20
      I=I+1
      IF(I.GT.N) GOTO 22
      S=C(I)-AMB-W(I-1)/S
```

```
      GOTO 18
20     I=I+2
      IF(I.LE.N) GOTO 16
22     IF(IG.GE.K) GOTO 24
      U=AMB
      GOTO 14
24     AL=AMB
      M=MIN0(IG,N)
      DO 26 I=K,M
26     E(I)=AMB
      GOTO 14
30     E(K)=AMB
      IF(AMB.GT.EPS) GOTO 15
      IF (NVEC.GE.K) NVEC=K-1
      NEV=K-1
C  ***********************************
C  *   CALCUL DES VECTEURS PROPRES   *
C  ***********************************
      L10=0
      DO 82 I=1,NVEC
      DO 44 J=1,N
      P(J)=0.0
      Q(J)=B(J)
      R(J)=C(J)-E(I)
44     Y(J)=1.0
      Y(N+1)=0.0
      Y(N+2)=0.0
      DO 50 J=1,NM1
      IF(ABS(R(J)).LT.ABS(B(J))) GOTO 46
      ULT=B(J)/R(J)
      VEC(J,I)=1.0
      GOTO 48
46     ULT=R(J)/B(J)
      R(J)=B(J)
      VEC(J,I)=-1.0
      T=R(J+1)
      R(J+1)=Q(J)
      Q(J)=T
      P(J)=Q(J+1)
      Q(J+1)=0.0
48     W(J)=ULT
      Q(J+1)=Q(J+1)-ULT*P(J)
      R(J+1)=R(J+1)-ULT*Q(J)
      IF(R(J).EQ.0.0) R(J)=EEE
50     CONTINUE
      IF(R(N).EQ.0.0) R(N)=EEE
51     DO 70 J=1,NM1
      IF(VEC(J,I).LT.0.0) GOTO 71
      Y(J+1)=Y(J+1)-W(J)*Y(J)
GOTO 70
      71     T=Y(J)
      Y(J)=Y(J+1)
      Y(J+1)=T-W(J)*Y(J+1)
70     CONTINUE
      DO 72 JJ=1,N
      K=N-JJ+1
72     Y(K)=(Y(K)-Y(K+1)*Q(K)-Y(K+2)*P(K))/R(K)
      L10=L10+1
      IF(L10.LE.1) GOTO 51
```

```
      DO 78 J=1,NM2
      T=0.0
      K=N-J-1
      M=K+1
      DO 76 KK=M,N
      L1=IT(KK)+K
76    T=T+A(L1)*Y(KK)
      L2=IT(K)+K
      TT=A(L2)*T
      DO 78 KK=M,N
      L3=IT(KK)+K
78    Y(KK)=Y(KK)-A(L3)*TT
      YYY=ABS(Y(1))
      DO 888 J=2,N
      IF(ABS(Y(J)).GT.YYY) YYY=Y(J)
888   CONTINUE
      DO 88 J=1,N
88    VEC(J,I)=Y(J)/YYY
      K=I-1
      DO 102 J=1,I
      Y(J)=0.0
      DO 102 JJ=1,N
102   Y(J)=Y(J)+VEC(JJ,I)*VEC(JJ,J)
      IF(I.EQ.1) GOTO 110
      DO 105 J=1,N
      DO 105 JJ=1,K
105   VEC(J,I)=VEC(J,I)-Y(JJ)*VEC(J,JJ)
110   TR=0.0
      DO 106 J=1,N
106   TR=TR+VEC(J,I)*VEC(J,I)
      TRA=0.0
      IF((TR/Y(I)).GT.EPS) TRA=SQRT(1/TR)
      DO 107 J=1,N
107   VEC(J,I)=VEC(J,I)*TRA
82    CONTINUE
C ***********************************
C *     IMPRESSION DES RESULTATS     *
C ***********************************
      IF(IQ.EQ.0) GOTO 120
      NR1=1
      NR2=8
      NR3=8
      IF(NVEC.LT.8) NR2=NVEC
121   WRITE(6,133) (J,J=NR1,NR3),(E(J),J=NR1,NR2)
      DO 122 I=1,N
122   WRITE(6,131) I,(VEC(I,J),J=NR1,NR2)
      IF(NVEC.LE.8) GOTO 120
      NR1=NR2+1
      NR2=NR1+7
      IF(NR2.GT.NVEC) NR2=NVEC
      IF(NR1.GT.NVEC) GOTO 120
      NR3=NR1+7
      GOTO 121
120   CONTINUE
      WRITE(6,134) E(1)
      CUM=0.0
      SYN=100.0/TEST
      EE=80.0/E(2)
      DO 135 I=2,NEV
```

```
      IF(CUM.GT.100.0) RETURN
      PC=E(I)*SYN
      CUM=CUM+PC
      II=INT(E(I)*EE+1.5)
      WRITE(6,132) I,E(I),PC,CUM,(IAST,K=1,II)
135   CONTINUE
131   FORMAT(1X,5HOBJET,I3,1H!,8(2X,F10.8,2X,1H!))
132   FORMAT(2H !,I3,3H ! ,F10.8,2(3H ! ,F7.3),2H !,I1,8(1H!,10I1))
133   FORMAT(1H1,15X,51HTABLEAU DES VALEURS PROPRES ET DES VECTEURS PROP
      RES,/1X,65(2H==),/1X,9H NUMERO !, 8(12H VAL PROPRE ,I2,1H!),/1X,65
      2(2H--),/1X,9HVECTEURS!,8(2X,F10.8,2X,1H!))
134   FORMAT(1H1,3X,32HPREMIERE VALEUR PROPRE (TRIVIALE),5X,7HVAL(1)=,F10
      1.8,/1H0,65(2H--),/1X,88H! NUM! VAL PROPRE ! POURCENT! CUMUL !*!
      2 HISTOGRAMME DES VALEURS PROPRES DE LA MATRICE,/1X,65(2H--))
      RETURN
      END
```