SEPTIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 28 MAI au 2 JUIN 1979

PROGRAMME CONVERSATIONNEL POUR L'ETUDE DU SIGNAL (PROCOPES). PRINCIPE D'ORGANISATION D'UN SYSTEME OUVERT ET INTERACTIF.

Jacques HAY, Jean-Paul BOISSEAU

Office National d'Etudes et de Recherches Aérospatiales (ONERA)
92320 Châtillon (FRANCE)

RESUME

- Si la puissance de calcul est le facteur prépondérant pour réaliser le traitement numérique de gros volumes de signaux suivant des procédures éprouvées, c'est davantage la facilité de programmation qui intéresse l'utilisateur lorsque l'on met au point une méthode nouvelle ou bien lorsque l'on souhaite réaliser le traitement d'un petit volume d'échantillons.

- Le système PROCOPES propose dans ce cas un exemple de solution : le traitement du signal s'adaptant bien à la programmation structurée, on peut écrire le programme sous forme d'une suite de macro-instructions gérées par un interpréteur.

- Le caractère interprétatif du langage permet une mise au point interactive dont l'aspect conversationnel est aidé par un pupitre de dialogue spécialement adapté. Le système est ouvert car il offre aux utilisateurs la possibilité de le compléter en écrivant -en FORTRAN- de nouvelles fonctions.

- Un exemple d'application est présenté et commenté et on évoque en conclusion l'intérêt de ce type d'organisation pour des matériels d'architecture moderne.

SUMMARY

PROCOPES : A CONVERSATIONAL PROGRAM FOR SIGNAL PROCESSING. ORGANIZATION OF AN INTERACTIVE AND VERSATILE SYSTEM.

Although high performance of a computer unit is the main feature to manage a "frozen" standard digital signal processing for a great deal of samples, scientists are often more concern with program writing conveniences.

In that case, PROCOPES system is one of the solutions : signal processing is well suited to structured programming and so it can be writen with a succession of macro-instructions controled by an interpreter.

Thanks to the interpretative character of language, an interactive debugging is possible ; for the special purpose of operator dialogue an appropriate console has been studied. Otherwise it is an open system because users can write FORTRAN subroutines which PROCOPES recognizes new functions.

An example is presented and commented ; as a conclusion the authors emphasize the interest of this type of system for modern architecture of interconnected computers.

PROGRAMME CONVERSATIONNEL POUR L'ETUDE DU SIGNAL (PROCOPES).
PRINCIPE D'ORGANISATION D'UN SYSTEME OUVERT ET INTERACTIF.

INTRODUCTION

PROCOPES est, à l'origine, un programme conversationnel destiné à gérer une chaîne de dépouillement constituée d'analyseurs en temps réel pilotés par un ordinateur. Grâce à une conception initiale très modulaire, le langage de commande assez pauvre et rigide dans les premières versions a pu évoluer vers un langage interprétatif plus puissant et plus souple, donc mieux adapté au "libre-service" de l'installation.

Pour présenter ce travail comme un outil de programmation en traitement du signal, il pourrait paraître suffisant de n'exposer que le langage PROCOPES ; en fait, il est nécessaire de présenter aussi l'organisation système de PROCOPES, car cette version reste très dépendante du calculateur sur lequel elle a été développée (CII - 10020 sous RBM).

De ce fait, l'intérêt de cette réalisation resterait-il restreint à cette chaîne de traitement si elle n'avait été l'occasion d'approfondir les divers problèmes que l'on rencontre pour créer ce type de logiciel. Ce travail devrait donc apporter une contribution utile à l'élaboration d'un système (ou "progiciel") portable, plus puissant mais de même finalité et dont la mise en oeuvre d'une version de base, puis son évolution, pourrait se développer favorablement dans le cadre d'une organisation analogue à celle adoptée pour MODULEF (IRIA) dans le domaine du calcul par éléments finis.

1 - PRINCIPE DU SYSTEME

Les buts étaient les suivants :

- Créer les principales fonctions utiles au traitement du signal en les organisant autour d'un module de FFT. (Ce qui conduira à privilégier dans les calculs sur tableaux la dimension 2^m).

- Faciliter l'éxécution conversationnelle en associant à ces fonctions des mnémoniques clairs, en choisissant une syntaxe pratique pour leurs arguments et en disposant d'un pupitre adapté au dialogue (Table Digistrand).

- Pouvoir à tout moment contrôler la valeur actuelle d'une variable (visualisation alphanumérique) ou d'un tableau (visualisation graphique).

- Enfin, en regroupant une suite de fonctions (macro-instructions), remplacer l'exécution conversationnelle par une exécution automatique selon un programme appelé ici "macro-programme".

Pour satisfaire ces objectifs, le langage de commande du mode conversationnel a été intégré dans un macrolangage de type interprétatif pour offrir la possibilité d'une exécution automatique. Dans la mise en oeuvre du système PROCOPES 4, l'exiguité de la configuration a conduit à une segmentation très poussée, nécessitant une gestion complexe d'échanges entre la mémoire et un disque ; son fonctionnement réel ne sera pas abordé ici.

La figure 1 donne le principe des deux modes : en conversationnel, l'opérateur envoie une commande au système grâce à un pupitre spécialement approprié (table Digistrand V.P.L.) en automatique le système va chercher une commande (macroinstruction) dans une table alphanumérique (code symbolique). Dans les deux cas, l'interpréteur analyse la commande, et "pointe" ensuite vers le SP correspondant qui s'exécute.

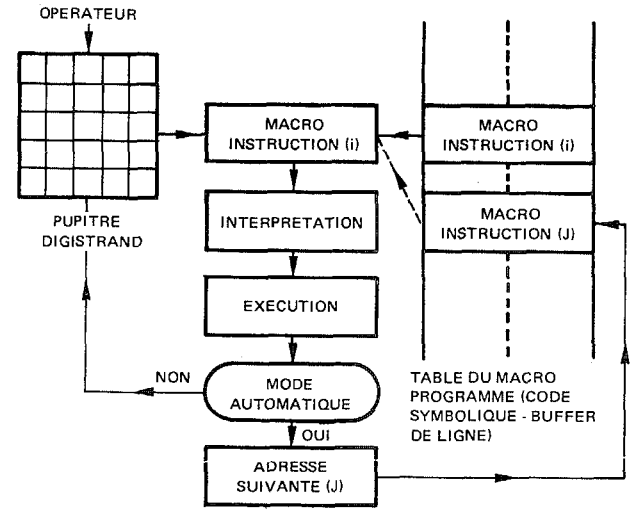

Fig.1-Principe des deux modes de fonctionnement de PROCOPES.

Il est évident que le caractère interprétatif de ce système qui doit "résoudre" une macro-instruction en partant de sa représentation alphanumérique chaque fois qu'elle est sollicitée dans le macroprogramme, lui confère en vitesse d'exécution des performances très médiocre, mais qui sont heureusement compensées par une grande facilité d'écriture et de mise au point appréciable dans bien des cas.

Cette facilité d'écriture est due en particulier aux moyens de gestion des arguments offerts par le système. On les évoque au paragraphe suivant.

2 - LES TYPES D'INFORMATION ET LEUR SYNTAXE

Les choix ont été guidés par la structure vectorielle fréquente en traitement du signal (notion de bloc) et par la nécessité de disposer d'une écriture à la fois souple et concise des arguments associés aux macroinstructions (liste, structure d'argument).

Aux six types d'information énumérés ci-dessous sont associées en mémoire leur représentation -dite interne- qui sera utilisée lors de l'exécution, ainsi que la représentation alphanumérique (ici codée EBCDIC) de leurs identificateurs qu'utilisera l'interpréteur pour les reconnaître.

Type 1 - Les Fonctions :

Elles seront évoquées au paragraphe 3. Leur syntaxe est de la forme :

X [XXX][XX], [arguments ou structure d'argument / VPL]

où [ ] désigne des caractères alphanumériques optionnels.

Ex. : ADD,A1,100,C2 qui recopie le "vecteur" A1 dans C2 après addition de 100 à toutes les composantes (V.P.L.).

Type 2 - Les Scalaires :

Ils comprennent :

- Les paramètres propres au système dont l'utili-

sateur peut modifier la valeur sans pouvoir les détruire (ex. : N = 2^n taille de la FFT)

- Les variables que l'on peut détruire ou créer.

Dans les deux cas, lors de leur création, il est impératif d'en définir le mode de représentation interne : entier ou flottant ; par contre l'opérateur peut, au pupitre de commande, en définir la valeur en format libre : 100, 0100, 100.00, +.1 +.1000 E+03 représentent les écritures de 100 affecté à un scalaire entier ou flottant.

Ex. : Après la commande de création ZE = 10,E qui donne à l'identificateur ZE la valeur 10 en entier (E), la commande ZE = 8.4 donnera à ZE la valeur 8.

Type 3 - Les blocs :

Ce sont des tableaux à une dimension de M valeurs flottantes ($1 \leqslant M \leqslant 2N$) où $N = 2^n$ est le paramètre définissant la taille de la FFT. ($N \leqslant 2048$).

La très petite configuration du calculateur a imposé d'en limiter le nombre et la taille. Si U symbolise les lettres A ou B ou C ou D, tandis que I, J, K symbolisent 1 ou 2, les blocs et sous-blocs de base sont désignés au système par U I[JK] où [JK]sont optionnels. Par exemple A1 est un bloc de base, A11, A12 s'en déduit par dichotomie de même que se déduisent A111 et A112 de A11.

Un bloc quelconque se définit par U I[JK] (L,M) où L et M désignent respectivement le rang du début et la taille.

Tous ces blocs peuvent être aussi symbolisés à l'aide d'identificateurs repérables par $. (Exemple : $B=A12(L,M) ou $B=A1(L',M)).

Le système dispose dans sa forme actuelle de huit blocs de base : A1,A2,B1,B2,C1,C2,D1,D2 et des sous blocs représentés ci-dessous à partir de A1 :

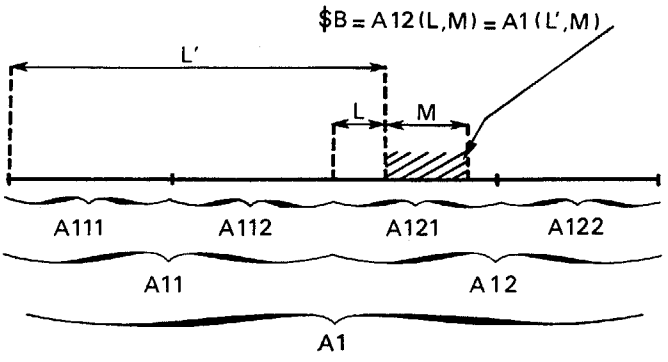

Fig.2-Exemple de répartition des blocs de base par dichotomie.

Type 4 - Les chaînes de caractères :

Elles sont repérées par deux apostrophes qui les encadrent, ex. : 'EDITION D''UN TEXTE'
On peut aussi les utiliser en faisant appel à un identificateur repérable par *X[XX]: après la commande *C ='EDITION D''UN TEXTE'

*C sera équivalent à la chaîne de caractères précédente.

Type 5 - Les listes :

Elles sont repérées par des parenthèses et peuvent englober les types 2 et 4 ; leurs identificateurs sont repérables par #

Ex. : ('MAX',MAX,'N=',1024)
repérable aussi par #L après la commande
#L=('MAX',MAX,'N=',1024)

Type 6 - Les structures d'argument :

C'est une suite d'arguments de types 2,3, 4,5 ou 6 (une structure peut en contenir une autre) séparés par des virgules et identifiables par | :

Ex. : A1(3,15),*C,#L,10
repérable aussi par |ST après la commande
|ST=A1(3,15),*C,#L,10 (x)

Dans les exemples précédents, sont apparues des commandes de création de différentes entités en plus des commandes de modification ; il leur correspond aussi une commande de destruction X[XXX]=% ; par exemple |ST=% supprime dans la table correspondante l'identificateur |ST et les valeurs associées par (x).

La notion de ligne de commande

En mode conversationnel l'exécution d'une commande est déclenchée par le caractère retour chariot noté NL (New Line), mais il est possible de définir une ligne de commandes séparées par des / : FOURIN/ADD,NESS,1/NESS=(NL).

Le caractère (NL) déclenchera l'exécution séquentielle de la ligne qui correspond ici à l'initialisation de la transformation de Fourier, l'addition de 1 à la variable NESS suivie par son affichage au pupitre de dialogue (commande NESS=).

Avant de citer les principales classes de Fonctions du système, on présente, figure 3, le pupitre de dialogue dont le logiciel a été spécialement développé pour PROCOPES : la table Digistrand correspond à une boîte à touches de Fonction servant aussi de clavier alphanumérique ; elle est associée à un écran qui par programme est partagé entre la visualisation des dialogues et des courbes. La figure 3a montre le principe d'utilisation de la table Digistrand : lors d'un pointé (X,Y) du crayon, le calculateur reconnaît la case correspondante (fig. 3b) à laquelle une table, gérée par le système, associe un caractère ou une chaîne de caractère (mnémonique de Fonction ou de paramètre du système). La figure 3b représente la grille associée à PROCOPES 4 ; la figure 3c donne un exemple de partage de l'écran entre (D), le dialogue et (V) la visualisation.

PROGRAMME CONVERSATIONNEL POUR L'ETUDE DU SIGNAL (PROCOPES).
PRINCIPE D'ORGANISATION D'UN SYSTEME OUVERT ET INTERACTIF.

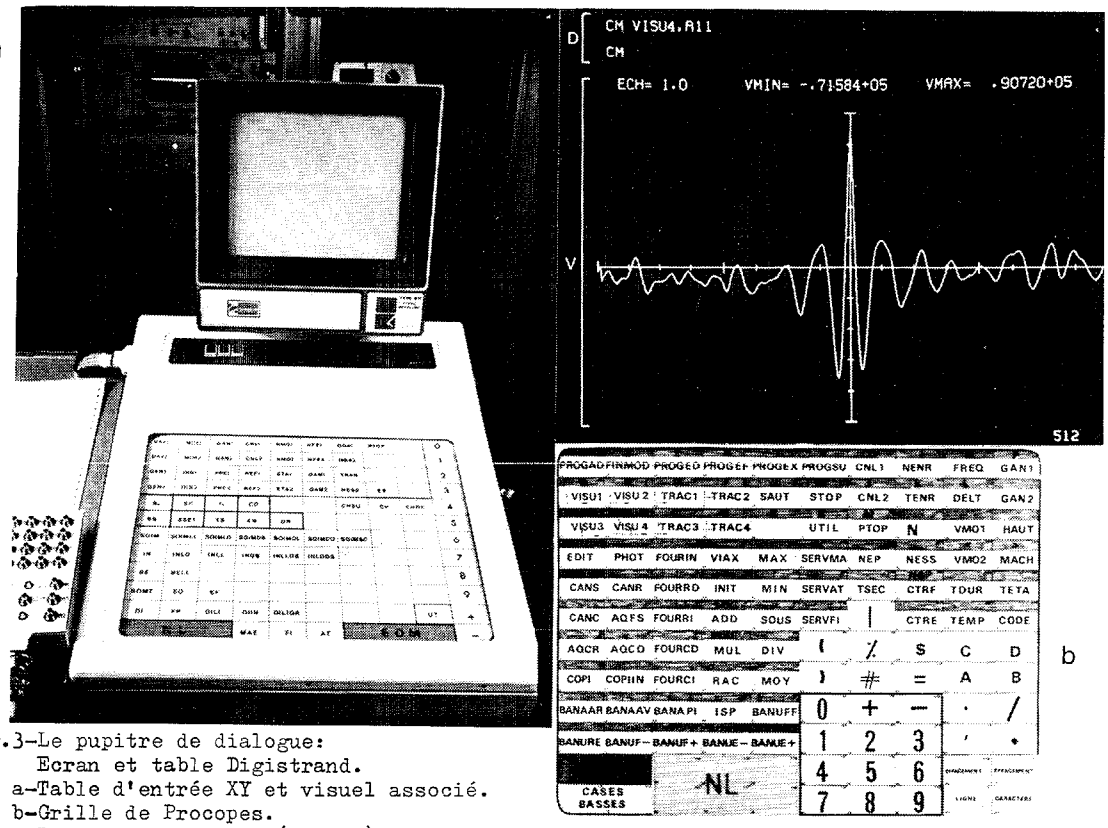

Fig.3-Le pupitre de dialogue:
Ecran et table Digistrand.
a-Table d'entrée XY et visuel associé.
b-Grille de Procopes.
c-Partage de l'écran.(D et V)

3 - LES FONCTIONS OU MACROINSTRUCTIONS DE PROCOPES

Pour pouvoir évoquer brièvement les fonctions dont les mnémoniques (ou codes) apparaissent sur la figure 3b, on les a regroupés par domaine d'utilité.

Les fonctions de gestion :

Elles permettent la gestion du système et son évolution en créant de nouveaux codes, des paramètres ou des cases de la table Digistrand (fonctions GENEXY), en offrant la possibilité de connaître l'état du système à tout instant (fonctions EDIT,X).

Elles permettent également la gestion des périphériques spécialement liés au traitement du signal : fonctions de gestion des dérouleurs analogiques et numériques consignant les signaux d'entrée ; fonctions de numérisation.

Les fonctions de traitement :

Elles sont limitées aux diverses formes de transformées de Fourier : directe ou inverse, sur des échantillons réels ou complexes.

Le système actuel, grâce aux fonctions de calcul (V.P.L.) permet d'en déduire par programme (écrit en macrolangage PROCOPES) les caractéristiques usuelles comme la corrélation ou la densité spectrale de puissance moyenne ; pour des traitements particuliers, on les écrit en FORTRAN sous forme de programme-utilisateur (V.P.L.).

Les fonctions arithmétiques sur blocs ou scalaires :

La syntaxe est de la forme : XXXX, a_1, $[a_2][,a_3]$ a_1 et a_2 sont des opérandes non commutatifs a_3 définit le rangement du résultat.

Par défaut $a_2 = a_1$; $a_3 = a_1$

Elles comprennent les quatre opérations (+)ADD, (-)SOUS, (x)MUL, (÷)DIV et des fonctions courantes du genre valeur moyenne (MOY) maximum et minimum (MAX,MIN), Racine Carrée (RAC) pour lesquelles, d'ailleurs, seuls les arguments a_1 et a_3 ont un sens.

Ex. :

ADD,B1(1),500,C1(1) qui fait C1(1)=B1(1)+500

ADD,B1,500,C1 qui fait C1(I)=B1(I)+500 I=1 à 2N

DIV, B1, C1 qui fait $B1(I) = \frac{B1(1)}{C1(I)}$ I= 1 à 2N

Les fonctions de représentations graphiques :

- La visualisation sur l'écran du pupitre de dialogue
Elle s'obtient par la fonction VISU I,[H/B], |X

où I = 1 à 4 est un sous-code propre au type d'axes, H ou B désignant la moitié haute ou basse de l'écran tandis que |X symbolise une structure d'argument particulière désignant à la fonction VISU le bloc à visualiser.

Exemple : La figure 3c correspond à la commande VISU 4, A11 (rappelée dans le coin supérieur gauche de l'écran qui visualise le dialogue) ; elle visualise une autocorrélation, résultat d'un macroprogramme (VPL) consigné dans le bloc A11.

Associée à VISU, la fonction PHOT reproduit l'image de l'écran sur table traçante.

- Le tracé s'obtient par la fonction TRAC.I, |X où I = 1 à 4. |X est une structure d'argument particulière permettant de nombreuses possibilités au niveau des titres, des commentaires, des renseignements propres aux axes, etc...

Les fonctions utilisateurs (F.U.) :

C'est grâce à ces fonctions que l'utilisateur peut traiter des signaux par des méthodes très particulières qu'il ne peut espérer trouver en tant que fonctions-standard d'un macrolangage qui doit nécessairement rester assez général. C'est aussi par l'intermédiaire de ces fonctions que l'ingénieur-système peut mettre au point facilement de nouvelles macroinstructions qu'il intégrera ensuite au macrolangage en leur choisissant un mnémonique approprié, la syntaxe des arguments ayant déjà été choisie et programmés au niveau du programme utilisateur :

En effet, ces Fonctions Utilisateurs écrites en Fortran sous forme de sous-programme peuvent communiquer avec PROCOPES à deux niveaux.

- Par la liste figée des arguments du S/P Fortran qui donne accès aux tables de PROCOPES ;

- Par des arguments associés à la commande de fonction-utilisateur que l'auteur peut traiter depuis son S/P Fortran en faisant appel à un sous-programme de "résolution" d'arguments ; la syntaxe est du type : UTIL n,|X
où n = 0,1,... est le numéro du S/P utilisateur et |X (optionnelle) désigne les arguments de la commande.

Ces fonctions associées aux fonctions de création de nouveaux codes contribuent à donner au système un caractère ouvert qui lui confère une bonne souplesse d'adaptation aux besoins ; cette propriété est fondamentale pour des logiciels de ce type.

Il existe enfin une dernière catégorie de fonctions qui interviennent dans l'écriture et l'exécution des macroprogrammes et qui seront évoquées au paragraphe suivant.

4 - MACROPROGRAMMES ET TRAITEMENT AUTOMATIQUE

En plus de l'usage conversationnel du système, l'utilisateur a besoin de créer une séquence de traitements exécutables automatiquement. Le système PROCOPES le lui permet, grâce à la possibilité d'écrire des programmes à partir des macro-instructions vues précédemment et appelés pour cela macroprogrammes.

Un macroprogramme est un ensemble de lignes de commandes étiquetées ou non, que l'on peut introduire en mémoire depuis le pupitre de dialogue (ou depuis d'autres périphériques : par exemple le lecteur de carte) sous le contrôle d'un éditeur de texte qui permet d'ajouter, d'effacer des lignes de commande et de les lister.

Pour pouvoir gérer (c'est-à-dire programmer) l'exécution des suites de macroinstructions en fonction de résultats intermédiaires, il a été créé une fonction SAUT, qui, après un test, branche sur des étiquettes de lignes. Suivant l'usage que l'on fait de ses arguments, on peut l'utiliser en branchement conditionnel ou inconditionnel, ainsi que dans la création de "sous-programme"

Pour illustrer les principales caractéristiques de ce macrolangage, on a choisi un exemple simple dont on donne (fig. 5) l'écriture éclatée et commentée (a, b, etc... y renvoient) ainsi que sa forme réelle organisée en lignes de commandes étiquetées (fig. 5). Il s'agit d'un programme qui numérise NENR portions de signal repérées par la variable ZONE ($1 \leqslant$ ZONE $\leqslant$ NENR). Une relation linéaire connue associe à ZONE l'heure moyenne de chaque portion définie par N échantillons. Il calcule ensuite le périodogramme P(v) (résultats dans le bloc A11). Là, un sous-programme écrit en Fortran par l'utilisateur (UTILOO) et pouvant disposer de ses propres arguments (ici A11 et B1 (ZONE)) recherche le maximum de P(v), calcule la puissance de la raie spectrale associée et la range dans B1 (ZONE). Enfin, le programme trace la puissance de cette raie en fonction du temps (commande TRAC1,IB) ; les arguments de la fonction de tracé lui sont fournis sous forme condensée dans IB, structure d'arguments définie en a, a_1, a_2, a_3 (fig. 5).

```
00   N=512/ZONE=0,E/NENR=70/*1="UTAC OCT 77"
01   #X=(TDUR,0,1,0-1,"SEC")
02    *P="PASSAGE"/*D=DISTANCE"
03    *V="VITESSE"/P=11,E/D=10,F/V=40,F
04    #P=(*P,P,2X,*D,D,2X,*V,V)
05    |B=B1(1,NENR),*1, #X, #P
06 1 STOP
07    ZONE=1/CANCOB/BANURE/COPIIN
08 2 COPI/FOURRD/MUL,A1,A1,A1
09    ADD,A11,A12,A11/UTILOO,A11,B1(ZONE)
10    ADD,ZONE,1/SAUT,ZONE,NENR,2
11    TDUR=0/ADD,NEP,N,TDUR /DIV,TDUR,FREQ
12    TRAC1,IB/SAUT,,,1
```

Fig. 4 - Programme sous sa forme condensée. (Affichage à l'écran)

CONCLUSION

L'intérêt de ce type d'organisation a pu être vérifié sur des problèmes concrets de traitement de signaux, c'est ainsi que des dépouillements demandant à la fois des fonctions classiques (mise à l'échelle, transformée de Fourier, visualisation, etc...) et des fonctions particulières ("Dédopplérisation", traitement d'antenne additive, estimation spectrale autoregressive...) (cf. [4]) ont pu être écrits rapidement, seule la partie spécifique ayant demandé une programmation FORTRAN pour créer les fonctions-utilisateur nécessaires.

L'enseignement majeur de ce travail aura été le suivant : que ce soit pour de gros systèmes disposant d'une bibliothèque des sous-programmes fondamentaux en traitement du signal (T.S.) ou bien que ce soit pour des systèmes modernes dont l'architecture a été définie spécialement pour le T.S. : calculateur maître associé à un calculateur spécialisé, il est possible de concevoir un logiciel presque portable -en Fortran par exemple- qui permettrait de créer un système analogue à PROCOPES sur des configurations informatiques très différentes. La seule portion de ce logiciel qui impliquerait l'écriture de quelques sous-programmes-assembleur serait celle qui assurerait l'interface entre l'interpréteur et l'appel soit à la bibliothèque T.S., soit au calculateur spécialisé.

Pour des traitements ou le temps d'exécution est critique, le programme en langage interprétatif peut être rapidement transcrit en Fortran pourvu que l'on dispose d'une gestion de fichier bien adaptée comme nous l'a montré récemment une transcription sur IRIS 80 de l'ensemble des programmes d'exploitation intervenant en métrologie de bruit non stationnaires [4].

PROGRAMME CONVERSATIONNEL POUR L'ETUDE DU SIGNAL (PROCOPES).
PRINCIPE D'ORGANISATION D'UN SYSTEME OUVERT ET INTERACTIF.

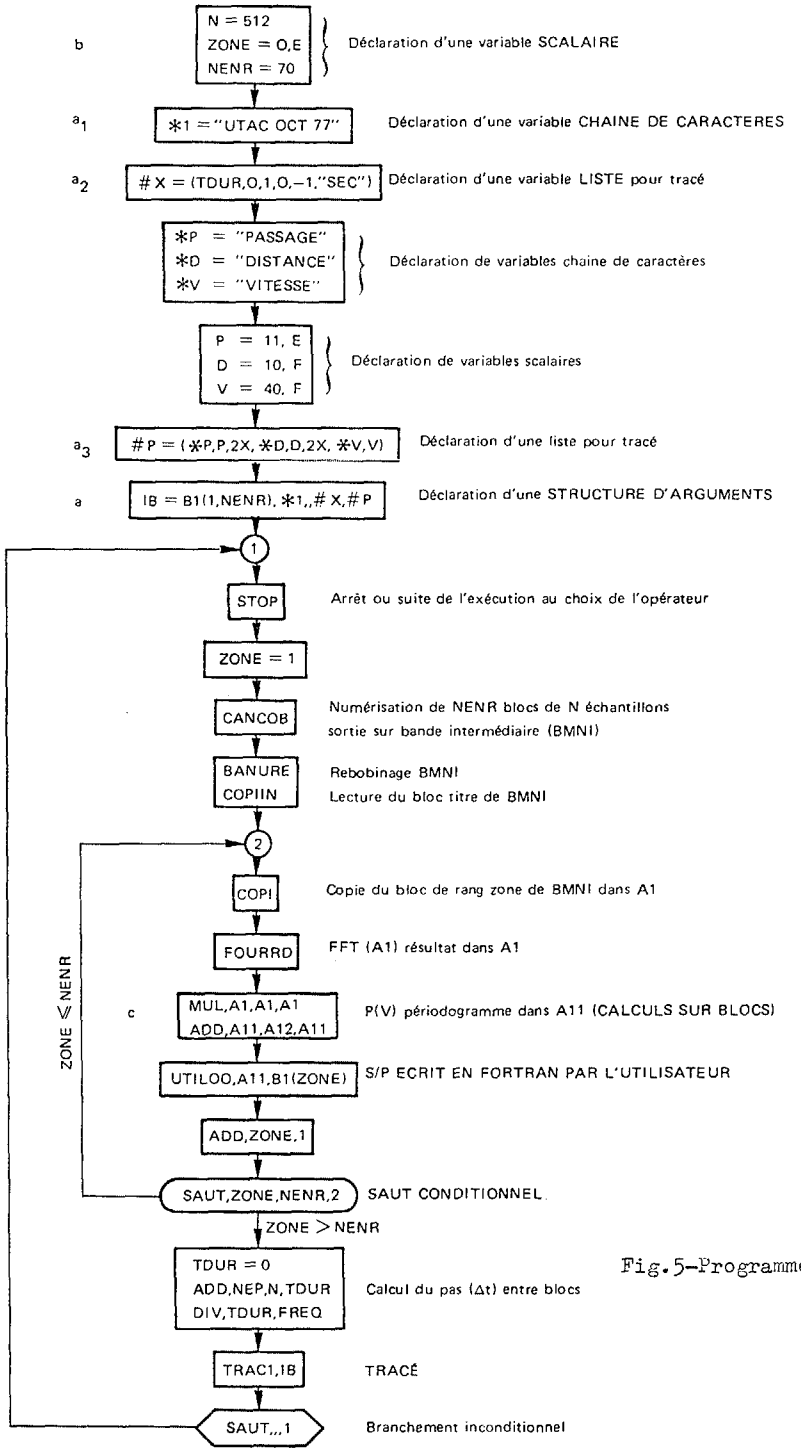

Fig.5-Programme éclaté et commenté.

BIBLIOGRAPHIE

[1] J.P. BOISSEAU, C. LE LURON, "PROCOPES 0" - RT 4/1887, document ONERA (1975) non publié.

[2] J.P. BOISSEAU, "PROCOPES" : évolution vers un langage de commande interprétatif. Thèse d'Université Paris-Sud-Orsay 24/11/78 et Note Technique ONERA, référence 1978-8.

[3] C. PAGE, "Analyse spectrale automatisée" - Thèse du 3ème cycle. Paris VII, 1975.

[4] Max ERNOULT, "La Dédopplérisation : un moyen d'améliorer l'imagerie des sources acoustiques animées d'un mouvement connu" - GRETSI 1979.