Vectorisation du calcul de la distribution de Wigner-Ville discrète sur Cyber 205

Vectorization of the computation of the discrete Wigner distribution on Cyber 205

Françoise PEYRIN

Laboratoire de Traitement du Signal et Ultrasons, Bât. 502, 20, avenue Albert-Einstein, 69621 VILLEURBANNE CEDEX

Françoise Peyrin a soutenu une thèse de 3e cycle en 1982 à l'INSA de Lyon qui porte sur la reconstruction et la visualisation d'images 3D en voludensitométrie par rayons X. Depuis 1981, elle est assistante au Laboratoire de Traitement du Signal et Ultrasons de l'INSA de Lyon et travaille dans le domaine du traitement numérique du signal et de l'image.

Robert GOUTTE

Laboratoire de Traitement du Signal et Ultrasons, Bât. 502, 20, avenue Albert-Einstein, 69621 VILLEURBANNE CEDEX

R. Goutte est Docteur ès Sciences, Professeur à l'INSA de Lyon et Directeur du Laboratoire de Traitement du Signal et d'Ultrasons à l'INSA de Lyon, ses domaines d'activités sont le traitement numérique du signal et de l'image.

François COURTEILLE

Control Data France, 27, cours des Petites-Écuries, BP n° 139, LOGNES, 77315 MARNE-LA-VALLÉE CEDEX 2

F. Courteille a obtenu le diplôme d'Ingénieur de l'INSA de Lyon, option informatique en 1977, Chef de projet dans le département des applications scientifiques de Control Data, ses domaines d'activités sont : génie logiciel, algorithmique numérique et non numérique vectorielle sur les super-calculateurs de Control Data Corporation. Il est Conseil en vectorisation et s'intéresse à l'intelligence artificielle.

RÉSUMÉ

Cet article décrit l'implantation d'un algorithme complètement vectorisé de calcul de la transformation de Wigner-Ville d'un signal sur Cyber 205. Le résultat est présenté sous la forme d'une image numérique. Les performances obtenues pour différentes tailles du signal d'entrée, ainsi que la comparaison entre les versions scalaires et vectorielles sont données.

MOTS CLÉS

Représentation temps-fréquence, transformation de Wigner-Ville, Cyber 205, algorithme, vectorisation.

SUMMARY

This paper describes the implementation of a vectorised algorithm allowing to compute the Wigner-Ville transform of a signal on the Cyber 205. The result is given as a discrete image. The performances obtained for different size of the input signal as well as the comparison between the scalar and vectorial versions of the algorithm are presented.

KEY WORDS

Time-frequency representation, Wigner distribution, Cyber 205, algorithm, vectorisation.

TABLE DES MATIÈRES

1. Introduction

Les représentations temps-fréquence permettant de caractériser un signal simultanément dans les domaines temporels et fréquentiels, jouent un rôle de plus en plus important pour l'étude des signaux non stationnaires. En particulier, la transformation de Wigner-Ville (TWV), introduite depuis de nombreuses années dans le domaine de la mécanique quantique [7] puis dans celui du traitement du signal [16], connaît depuis quelques temps un regain d'intérêt. En effet, elle apparaît comme une représentation temps-fréquence de base possédant des propriétés mathématiques intéressantes [3]. Un grand nombre de travaux ont permis d'apprécier son comportement théorique vis-à-vis de certaines classes de signaux, son potentiel d'applications et ses particularités par rapport à d'autres représentations de ce type [1, 4, 7, 13]. De plus, son application à des signaux physiques et son interprétation dans le cas de problèmes pratiques ont fait l'objet d'études récentes [8, 9, 12, 14].

La TWV transforme un signal monodimensionnel en une représentation bidimensionnelle. Lorsque la taille du signal devient importante et *a fortiori* lorsque l'étude nécessite le traitement d'une série de signaux, il est important de disposer d'un outil puissant pour la calculer. Pour cela, différentes solutions peuvent être envisagées; en particulier un processeur spécialisé à architecture systolique a été proposé à cet effet [6]. Dans cet article, nous décrivons l'implantation du calcul de la TWV sur un ordinateur vectoriel, le Cyber 205 de Control Data, permettant de disposer à la fois d'une capacité mémoire importante et de performances de calcul intéressantes.

Dans une première partie, nous rappelons la définition de la TWV continue et les problèmes liés à sa discrétisation. Après avoir donné l'expression de la version discrète retenue (TWVD) nous exposons le principe de son calcul. Afin d'implanter le programme sur le Cyber 205 et obtenir de bonne performances, il est nécessaire de vectoriser l'algorithme, c'est-à-dire d'adapter le programme le mieux possible à la structure vectorielle de la machine. Nous présentons donc ensuite des généralités sur les architectures vectorielles puis les caractéristiques du Cyber 205 et quelques principes de base de la vectorisation. Après avoir analysé le problème du calcul de la TWVD et implanté une version scalaire de ce programme, nous décrivons la manière dont cet algorithme a été vectorisé. Enfin, nous présentons les performances atteintes par le programme vectorisé, les gains obtenus par rapport au calcul scalaire et des exemples d'éxécution.

2. La transformation de Wigner-Ville (TWV)

2.1. La transformation de Wigner-Ville continue

La TWV continue associe à un signal temporel complexe d'énergie finie $z(t)$ une fonction réelle $W_z(t, \nu)$

du temps t et de la fréquence ν définie par l'une des deux définitions équivalentes suivantes :

(1) $W_z(t, \nu) = \int z(t+\theta/2)\, z^*(t-\theta/2) \exp(-2i\pi\nu\theta)\, d\theta$

(2) $W_z(t, \nu) = \int Z(\nu+\eta/2)\, Z^*(\nu-\eta/2) \exp(2i\pi t\eta)\, d\eta$

où $Z(\nu)$ représente la transformée de Fourier de $z(t)$.

Cette fonction $W_z(t, \nu)$, appelée Distribution de Wigner-Ville (DWV), est une représentation énergétique du signal.

Il nous paraît important de mettre en parallèle les définitions de la TWV à partir du signal temporel $z(t)$ [définition (1)] et à partir de son spectre $Z(\nu)$ [définition (2)]. En effet, leur similitude fait pressentir les rôles duaux du temps t et de la fréquence ν dans cette représentation. Ceci est mis en évidence par l'examen des propriétés de la TWV détaillées par exemple dans [3]. Nous rappelons ici brièvement quelques-unes de ces propriétés. Par projection sur l'axe des temps (resp. des fréquences), on obtient la puissance instantanée du signal (resp. la densité spectrale). Il en résulte que si le signal est de durée finie T (resp. à spectre limité de largeur de bande B), le support de sa DWV par rapport à la variable temporelle (resp. fréquentielle) est T (resp. B). La double somme de la DWV sur le temps et la fréquence est donc égale à l'énergie du signal, ce qui explique que l'on tende à interpréter la DWV comme la densité d'énergie du signal dans le plan temps-fréquence. Toutefois, cette interprétation ne serait correcte que si la DWV était une fonction positive. Ceci n'étant pas le cas de façon générale, la DWV pouvant être localement négative. Son moment par rapport au temps (resp. la fréquence) permet de déterminer la fréquence instantanée du signal (resp. son retard de groupe). Signalons enfin que la TWV est une transformation non linéaire, mais réversible.

2.2. La transformation de Wigner-Ville discrète (TWVD)

Une première version de la transformation de Wigner-Ville dans le cas de signaux temporels discrets a été définie et étudiée par Claasen et Mecklenbrauker [3 (part II)]. Toutefois, il est apparu que toutes les propriétés de la TWV continue n'étaient pas préservées par discrétisation du fait de problèmes de recouvrements. Ceci a conduit Chan [2] ainsi que les auteurs précédents [5] à proposer de nouvelles définitions permettant de conserver les propriétés de la TWV continue.

Dans le but de faire un choix parmi les différentes définitions discrètes, nous avons utilisé une autre approche consistant à étudier les effets de l'échantillonnage du signal ou de son spectre sur cette transformation [15]. Nous avons montré que la DWV d'un signal temporel échantillonné au pas d, est elle-même échantillonnée temporellement au pas $d/2$ et périodisée en fréquence à la période $1/2d$ à une alternance de signe près. L'échantillonnage du spectre a une incidence analogue sur la DWV en permutant les rôles du temps et de la fréquence. Ces remarques nous ont permis d'introduire une version de la TWV à la fois discrète en temps et en fréquence, définie par l'une des deux définitions équivalentes :

$$(4)\quad W(n, m) = \frac{1}{2N} \sum_{k=0}^{N-1} z_k z^*_{n-k} \exp\frac{-i\pi m}{N}(2k-n)$$

$$(5)\quad W(n, m) = \frac{1}{2N} \sum_{l=0}^{N-1} Z_l Z^*_{m-l} \exp\frac{i\pi n}{N}(2l-m)$$

où N est le nombre d'échantillons, $z_0, z_1, \ldots, z_{N-1}$ (resp. $Z_0, Z_1, \ldots, Z_{n-1}$) est le signal (resp. le spectre) discret et où les indices sont considérés modulo N.

Considérant cette définition, la plupart des propriétés de la TWV continue sont préservées. Toutefois, les problèmes de recouvrements subsistent sauf si les conditions suivantes sont réalisées :

$$(6)\quad z_k = 0 \quad \text{pour} \quad k \geqq N/2$$

$$(7)\quad Z_l = 0 \quad \text{pour} \quad l \geqq N/2$$

La première condition (6) n'est pas restrictive si le signal est de durée finie, la seconde (7) impose que le signal soit suréchantillonné au moins par un facteur deux. En l'absence de recouvrement, la DWV discrète (DWVD) est reliée à la DWV du signal continu par la relation :

$$(8)\quad W(n, m) = \frac{1}{4Nd} W_z\left(\frac{nd}{2}, \frac{m}{2Nd}\right)$$

pour $0 \leqq n \leqq N-1$ et $0 \leqq m \leqq N-1$.

La DWVD est ainsi définie avec un pas d'échantillonnage en temps (resp. en fréquence) deux fois plus fin que celui du signal (resp. du spectre).

2.3. Notion de signal analytique

Dans le cas d'applications physiques, les signaux à traiter sont réels. Il a été prouvé qu'il est alors plus intéressant de travailler sur le signal analytique [16] associé au signal réel. Si $x(t)$ représente un signal réel, le signal analytique $z(t)$ qui lui est associé est défini par :

$$(9)\quad z(t) = x(t) - i\mathcal{H}\, x(t)$$

ou par la propriété fréquentielle équivalente :

$$(10)\quad Z(\nu) = \begin{cases} 2X(\nu) & \text{si } \nu > 0 \\ X(0) & \text{si } \nu = 0 \\ 0 & \text{si } \nu < 0 \end{cases}$$

où $\mathscr{H}$ représente la transformation de Hilbert et, $Z(\nu)$ et $X(\nu)$ les transformées de Fourier respectives de $z(t)$ et $x(t)$.

Toute l'information présente dans le signal réel se retrouve contenue dans le signal analytique sous une forme différente. Pour l'application de la TWV, cette notion présente au moins deux avantages :

– D'une part, la DWV du signal analytique a une structure plus simple que celle du signal réel. En effet, dans cette dernière, on retrouve la DWV du signal analytique, symétrisée (fréquences positives et négatives) plus un terme croisé dû à l'interaction entre les fréquences positives et négatives qui n'apporte pas d'information supplémentaire, mais ne peut que rendre l'interprétation de la DWV plus difficile [10].

– D'autre part, le spectre du signal analytique a une largeur de bande deux fois plus faible que celui du signal réel. Lorsque le signal réel est échantillonné à la fréquence de Shannon, le signal analytique est suréchantillonné par un facteur deux. Ceci permet de s'affranchir des problèmes de recouvrements fréquentiels intervenant dans la DWVD.

Dans le cas discret, nous retiendrons l'équivalent discret de la relation (10), c'est-à-dire :

$$(11)\quad Z_k = \begin{cases} 2X_k & \text{si } 1 \leqq k \leqq \dfrac{N}{2}-1 \\ X_k & \text{si } k=0 \text{ ou } k=N/2 \\ 0 & \text{si } \dfrac{N}{2}+1 \leqq k \leqq N-1 \end{cases}$$

où X_k et Z_k $(k=0, \ldots, N-1)$ sont les transformées de Fourier Discrètes respectivement du signal réel discret $x_0, x_1, \ldots, x_{N-1}$ et du signal analytique discret $z_0, z_1, \ldots, z_{N-1}$. Cette définition revient à utiliser une version discrète et périodisée de l'échelon unité, la valeur 1/2 étant affectée aux points de discontinuités.

2.4. Principe du calcul

Le programme permettant le calcul de la DWV comporte essentiellement trois modules :

(*a*) calcul du signal analytique;

(*b*) calcul de la TWVD d'un signal complexe;

(*c*) organisation des résultats.

Pour éviter tout problème de recouvrement dans le cas d'un signal d'entrée réel, le calcul de la TWVD sera effectué sur le signal analytique bordé de zéros, associé au signal réel. Le calcul du signal analytique est facilement réalisé en utilisant la définition (11) et la transformation de Fourier Discrète (TFD). Si le signal d'entrée réel est sur N/2 points, cette partie nécessite approximativement $N/2(\log_2 N)$ multiplications complexes et $N(\log_2 N-1)$ additions complexes lorsque l'algorithme FFT radix 2 est utilisé.

Le deuxième module calcule la TWVD d'un signal complexe sur N points. Ce calcul repose sur la définition (4) qui peut être décomposée selon que n est pair ou impair en :

$$(12)\quad W(2p, m)=\frac{1}{2N}\sum_{k=0}^{N-1} z_{p+k} z^*_{p-k} \times \exp\left(\frac{-2i\pi mk}{N}\right)$$

et :

$$(13)\quad W(2p+1, m)=\frac{1}{2N}\sum_{k=0}^{N-1} z_{p+k} z^*_{p-k+1} \times \exp\left(\frac{-2i\pi mk}{N}\right)\exp\left(\frac{i\pi m}{N}\right)$$

où p varie de 0 à $N/2-1$ et où tous les indices sur z sont considérés modulo N.

Le calcul peut donc s'exprimer à l'aide de transformées de Fourier Discrètes (TFD) qui seront post-multipliées par un terme en exponentiel dans le cas des lignes d'indices impairs. Plus précisément, on peut donner les schémas suivants :

Calcul des lignes paires $(n=2p)$

```
Pour p=0, N/2-1 faire
  | Pour k=0, N-1 faire
  | r_k = z_{p+k (mod N)} z*_{p-k (mod N)}
  | fin pour
    TFD de { r_k/k=0, N-1 }
  | Pour m=0, N-1 faire
  | W(2p, m) = r_m
  | fin pour
fin pour
```

Calcul des lignes impaires $(n=2p+1)$

```
Pour p=0, N/2-1 faire
  | Pour k=0, N-1 faire
  | r_k = z_{p+k (mod N)} z*_{p-k+1 (mod N)}
  | fin pour
    TFD de { r_k/k=0, N-1 }
  | Pour m=0, N-1 faire
  | W(2p+1, m) = r_m exp(i π m / N)
  | fin pour
fin pour
```

L'algorithme peut être légèrement optimisé en exploitant la structure des vecteurs $\{r_k/k=0, N-1\}$. En effet, on a pour les lignes d'indices pairs :

$$(14)\qquad r_{N-k}=r_k^* \qquad \text{pour} \quad 0 \leqq k \leqq N-1$$

et pour les lignes d'indices impairs :

$$(15)\quad \begin{cases} r_{N+1-k}=r_k^* & \text{pour} \quad 2 \leqq k \leqq N-1 \\ r_1=r_0^* \end{cases}$$

Il suffit donc de calculer la moitié des produits d'éléments de z, l'autre moitié s'obtenant par conjugaison.

Le module de calcul de la DWV nécessitant N appels à la FFT, utilise donc approximativement N^2 $(\log_2 N+3)/2$ multiplications complexes et $N^2 \log_2 N$ additions complexes.

Le dernier module concerne l'organisation des résultats. La TWVD d'un signal complexe discret sur N points étant une matrice réelle de dimension $N \times N$, il apparaît intéressant de considérer le résultat comme une image numérique. Celle-ci pourra alors être visualisée et exploitée sur un système de traitement d'images. Pour cela, les valeurs de la TWVD sont recadrées linéairement entre 0 et 255 et restructurées afin d'être stockées dans un fichier compatible avec le système de traitement d'images.

Avant de détailler la mise en œuvre de cet algorithme sur un calculateur vectoriel (Cyber 205), nous rappelons quelques généralités sur l'architecture de ce type de calculateur, décrivons brièvement les caractéristiques du Cyber 205 et quelques principes de base de vectorisation [11].

3. Les architectures vectorielles

3.1. Généralités sur les calculateurs vectoriels

En réponse aux besoins croissants en puissance de calcul que nécessitent notamment les domaines de la simulation numérique et du traitement du signal que, seuls les progrès de la technologie des circuits intégrés et de l'algorithmique ne suffisent pas à satisfaire, des structures nouvelles de calculateurs ont été développées. Ces nouvelles architectures mettent en œuvre le parallélisme. Ce terme désigne un ensemble de techniques variées qui ont en commun de maintenir simultanément actifs, un nombre élevé d'organes du système informatique. La classification des architectures rapides étant délicate, nous nous contenterons de présenter ici les architectures vectorielles qui sont les plus couramment commercialisées.

Les ordinateurs vectoriels atteignent des puissances de calcul élevées grâce à des instructions qui opèrent sur des ensembles de nombres, les vecteurs, au lieu de nombres isolés, les scalaires. Dans la suite de cet article, nous distinguerons ainsi les « machines vectorielles » des « machines séquentielles ou scalaires ». Alors que les performances des machines séquentielles sont le plus souvent mesurées en mips (millions d'instructions par seconde) celles des machines vectorielles sont exprimées en mégaflops (millions d'opérations flottantes par seconde) : produire un résultat pour une machine scalaire, nécessite l'exécution de plusieurs instructions; à l'opposé, une seule instruction vectorielle pourra produire plusieurs milliers de résultats. La structure des processeurs vectoriels repose essentiellement sur la combinaison de deux architectures de base : pipe-line et SIMD (Single Instruction stream Multiple Data stream).

Une unité est dite pipe-line lorsqu'elle peut prendre en charge une nouvelle opération à chaque période d'horloge (appelée encore cycle) alors que l'exécution complète de chaque opération en elle-même nécessite plusieurs cycles (*fig.* 1). Donc globalement tout se passe comme si l'unité exécutait une opération par cycle. Pour un ordinateur séquentiel, le temps nécessaire pour effectuer N opérations est Np cycles lorsque chaque opération nécessite p cycles. Par contre, pour un calculateur vectoriel pipe-line, le même traitement utilise seulement $p+(N-1)$ cycles. Pour les vecteurs « courts », c'est-à-dire comportant un faible nombre d'éléments (N petit devant p) le temps nécessaire à l'apparition du premier résultat (p cycles) appelé temps d'initialisation, domine cette expression. Au contraire, pour les vecteurs longs, le débit est proche de la valeur asymptotique : un résultat à chaque cycle.

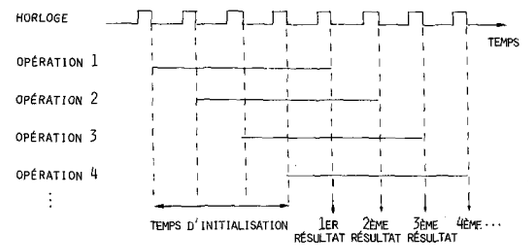


Fig. 1. – Schéma de principe du déroulement des *opérations* dans le temps sur une unité pipe-line. L'opération représentée dure quatre cycles ($p=4$). Une fois le pipe-line initialisé, au bout de p cycles, on obtient un résultat par cycle.

Dans un processeur SIMD, plusieurs opérations identiques sont exécutées simultanément par M unités de traitement qui travaillent chacune sur une opération (*fig.* 2). Dans le cas où la longueur N du vecteur traité est inférieure ou égale à M, le processeur effectue une opération vectorielle dans le temps T nécessaire au calcul d'une seule opération sur un calculateur séquentiel. Dans le cas opposé où N est strictement supérieur à M, le vecteur sera décomposé en « tranches » de M éléments et la durée de l'opération vectorielle est $[N/M] \times T$ au lieu de NT sur un calculateur séquentiel (le symbole [ ] désignant « l'entier supérieur le plus proche »).

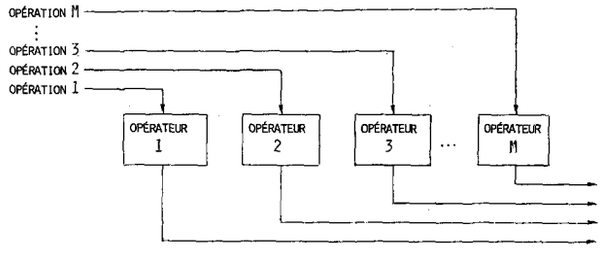


Fig. 2. – Schéma de principe d'un processeur SIMD. M unités de traitement travaillent simultanément sur M opérations identiques.

3.2. Caractéristiques du Cyber 205

Dans le cas du Cyber 205 (*fig.* 3), les deux approches pipe-line et SIMD sont utilisées.

Les unités du processeur vectoriel utilisent la technique du pipe-line aussi bien pour les unités de calcul

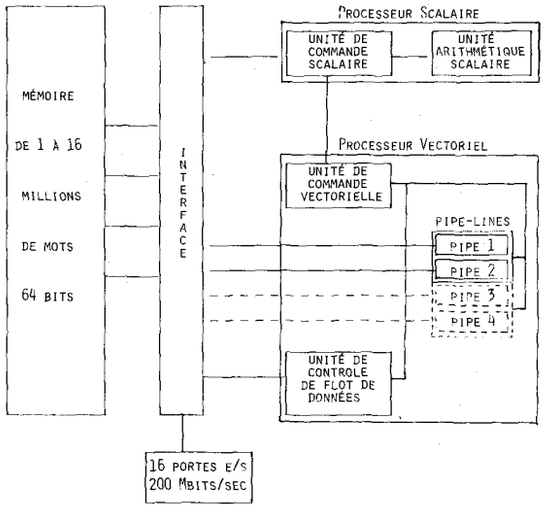


Fig. 3. – Schéma synoptique de l'architecture du Cyber 205.

arithmétique que pour les unités de lecture/écriture en mémoire. De plus, le Cyber 205 existe dans des configurations à 1, 2 ou 4 « pipes » (unités de calcul vectoriel), ce qui constitue un emprunt à l'approche SIMD. Le Cyber 205 possède aussi un processeur scalaire dont la puissance atteint 50 mips (millions d'instructions par seconde).

Le processeur scalaire décode les instructions provenant de la mémoire : si elles sont de type scalaire, elles sont exécutées, sinon elles sont transmises au processeur vectoriel.

Les pipe-lines sont directement connectés à la mémoire centrale et peuvent produire de 200 à 800 Mflops (millions d'opérations virgule flottante par seconde) selon que l'on dispose de deux ou quatre pipe-lines. La machine utilisée pour cette étude située à Marne-la-Vallée possède deux pipe-lines et une capacité mémoire de 4 millions de mots de 64 bits. Pour satisfaire les débits de données très élevées (25,6 ou 51,2 gigabits par seconde) que nécessitent les unités de calcul vectoriel, la mémoire est aussi organisée suivant le principe du pipe-line.

Le système d'exploitation à mémoire virtuelle (VSOS) met à la disposition de l'utilisateur un espace virtuel de plus de 2 200 milliards de mots, ce qui simplifie l'écriture des applications.

De par son architecture pipe-line, le Cyber 205 est spécialement adapté au traitement de vecteurs, définis comme suites d'éléments contigus en mémoire virtuelle. Il possède un jeu important d'instructions vectorielles accessibles par l'intermédiaire des extensions apportées au FORTRAN ANSI 77. On peut illustrer le gain apporté par le calcul vectoriel : une addition vectorielle de longueur N utilise $51+(N/2)$ cycles sur le Cyber 205 de Marne-la-Vallée alors que le même traitement en scalaire en utiliserait 36 N. Dans ce cas, on obtient asymptotiquement un rapport du temps scalaire sur le temps vectoriel de 72. Il est évident que plus la taille du vecteur est importante, dans la limite admissible (c'est-à-dire 65 535 éléments), plus le gain apporté par la vectorisation est grand.

3.3. Introduction a la vectorisation [11]

Rappelons d'abord qu'une instruction vectorielle est une instruction câblée opérant sur des vecteurs et produisant un vecteur, les vecteurs sur le Cyber 205 étant définis comme suites d'éléments contigus en mémoire virtuelle. Ces éléments peuvent être des bits, des octets, des demi-mots de 32 bits ou des mots de 64 bits. Un vecteur est caractérisé par son type (BIT, REAL, COMPLEX, ...), la donnée de son premier élément et sa longueur.

La vectorisation consiste, soit en la conception et l'écriture, soit en la transformation de programmes de telle façon que le plus grand nombre d'opérations arithmériques et logiques soit exécuté comme instructions vectorielles, ceci afin d'exploiter au maximum les performances de la machine.

La production d'instructions vectorielles peut être obtenue, soit en écrivant le programme en FORTRAN conventionnel, puis en utilisant la « vectorisation automatique », soit en écrivant explicitement le programme dans une syntaxe vectorielle.

La première solution consiste à utiliser l'option de vectorisation automatique du compilateur FORTRAN. Celui-ci reconnaît un certain nombre de structures « vectorisables » et génère les instructions vectorielles correspondantes. A titre d'exemple, la boucle :

$$(16)\qquad \begin{cases} \text{D0 1 I} = 1,512 \\ \text{1 A (I)} = \text{B (I)} + \text{C (I)} \end{cases}$$

est automatiquement traduite en une addition vectorielle de longueur 512 (génération d'une seule instruction en machine). Les structures syntaxiques vectorisables sont des boucles D0 satisfaisant un certain nombre de conditions [18]. En général, les opérations arithmétiques et logiques sur des tableaux sont vectorisables.

Toutefois, l'existence de certaines structures dans les boucles D0 (tests, branchement, non-contiguité des données, récurrences, appel à des sous-programmes, ...) inhibe la vectorisation automatique.

Par exemple, considérons les deux boucles suivantes :

$$(17)\qquad \begin{cases} \text{D0 2 I} = 1,100 \\ \text{IF ((A (I) . GT . B (I)) A (I)} = 0 \\ \text{2B (I)} = \text{A (I)} + \text{C (I)} \end{cases}$$

ou

$$(18)\qquad \begin{cases} \text{DATA J/4,2,1,5,3/} \\ \text{D0 3 I} = 1,5 \\ \text{3A [J (I)]} = \text{B (I)} \end{cases}$$

La boucle (17) comportant un test (IF) n'est pas vectorisée automatiquement par le compilateur. Il en est de même pour la boucle (18), les éléments du tableau A intervenant dans cette boucle [c'est-à-dire A (4), A (2), A (1), A (5), A (3)] n'étant pas contigus en mémoire virtuelle.

Une seconde solution consiste à vectoriser explicitement le programme en utilisant une syntaxe vectorielle. Sur le Cyber 205, tout le jeu d'instructions vectorielles de la machine est utilisable grâce au compilateur FORTRAN 200 (extension du FORTRAN ANSI 78). La notation X (I; N) désigne l'ensemble des N éléments consécutifs à partir de l'adresse X (I), soit X (I), X (I+1), . . ., X (I+N−1). Avec ces notations, la boucle (16) s'écrit simplement :

$$A(1;\ 512) = B(1;\ 512) + C(1;\ 512)$$

L'utilisation de cette syntaxe permet de vectoriser des boucles qui n'étaient pas reconnues vectorisables par le compilateur. Ainsi, la boucle (17) peut être vectorisée en utilisant la structure de contrôle vectorielle « WHERE » :

$$\text{WHERE}\,[A(1;\ 100)\,.\,\text{GT}\,.\,B(1;\ 100)]\ A(1;\ 100) = 0.$$
$$B(1;\ 100) = A(1;\ 100) + C(1;\ 100)$$

D'autres structures se vectorisent en modifiant localement l'ordre des données ou en créant des vecteurs temporaires. Pour cela, on peut utiliser des instructions du type SCATTER/GATHER ou COMPRESS/MERGE. Par exemple, la boucle (18) peut être vectorisée en rassemblant les éléments de B sous la forme d'un vecteur, l'ordre des éléments étant défini par le tableau d'indice J, grâce à l'instruction SCATTER :

$$A(1;\ 5) = \text{Q8VSCATR}\,(B(1;\ 5),\ J(1;\ 5);\ A(1,5))$$

Toutefois, il arrive qu'une réorganisation locale des données ne permette pas d'atteindre des performances optimales. Dans ce cas, on est amené à réorganiser globalement la structure de données, à modifier l'algorithme et éventuellement la méthode mathématique envisagée. Dans ce cas, le problème doit être « pensé » entièrement en termes de vecteur, c'est-à-dire que les différentes étapes de la création du programme doivent être abordées en fonction du fait que les traitements se font sur des ensembles d'éléments au lieu d'éléments.

4. Vectorisation de l'algorithme

4.1. Algorithme scalaire

Dans un premier temps, l'algorithme scalaire, tel qu'il a été décrit dans le paragraphe 2.4, a été mis en œuvre sur le Cyber 205 en FORTRAN ANSI 78.

Compte tenu de la description des opérations arithmétiques du Cyber 205, la partie calcul du signal analytique nécessite de l'ordre de $N(5\log_2 N - 2)$ opérations réelles et la partie calcul de la TWVD de l'ordre de $N^2(5\log_2 N + 24) + 56{,}5\,N$ opérations réelles. Quant à la partie concernant le recadrage linéaire du tableau, on peut l'estimer à environ $6\,N^2$ opérations réelles.

Tableau I

TSIG, DWV, TREC, temps d'exécution exprimés en millisecondes respectifs des modules de calcul du signal analytique, de la transformée de Wigner-Ville et du recadrage linéaire, en utilisant l'algorithme scalaire.

N . .	64	128	256	512	1 024
TSIG . . .	1,2	2,4	5,1	10,6	21,8
TWV . . .	94,2	385,9	1 592,1	6 572,6	26 940
TREC . .	9,6	38,1	152,4	619,7	2 478,8

Les temps d'exécution du calcul du signal analytique TSIG, du calcul de la transformée de Wigner-Ville TWV et du recadrage linéaire TREC, exprimés en millisecondes, sont portés dans le tableau I pour différentes valeurs de N. Le tableau II donne les vitesses d'exécution respectives MSIG, MWV, MREC exprimées en mégaflops (million d'opérations flottantes par seconde) pour ces trois parties.

Tableau II

Vitesse d'exécution (Mflops), arrondie par défaut, des trois parties du programme en utilisant l'algorithme scalaire.

N.	64	128	256	512	1 024
MSIG.	1,5	1,7	1,9	2,1	2,3
MWV.	2,4	2,5	2,6	2,8	2,9
MREC	2,5	2,5	2,5	2,5	2,5

4.2. Analyse du problème

L'effort de vectorisation doit principalement être produit sur les modules les plus consommateurs en temps UC (unité centrale). Généralement, 90% du temps UC sont passés dans 10% du code. Ici, l'examen des complexités des différentes parties du programme (*cf.* §4.1) montre que le calcul de la TWVD est prépondérant. Notre première priorité sera donc de vectoriser ce calcul. Les principaux problèmes sont dus au fait que :

– Les calculs prépondérants dans cet algorithme sont les FFT qui ne se vectorisent pas automatiquement.

– La plupart des calculs sont faits en arithmétique complexe. La vectorisation automatique d'une opération vectorielle complexe introduit des opérations supplémentaires du fait du stockage des nombres complexes en machine. En effet, un nombre complexe est assimilé à un couple de réels (partie réelle, partie

imaginaire). Un vecteur complexe de longueur N est alors représenté par un vecteur réel de longueur 2N où alternent partie réelle et partie imaginaire. Toute opération vectorielle complexe nécessite ainsi le découpage préalable du vecteur complexe en deux vecteurs réels avant que l'opération proprement dite ne soit effectuée et que les résultats ne soient fusionnés dans le vecteur complexe final.

Afin d'obtenir des performances intéressantes, il apparaît donc nécessaire de :

– modifier notre mode de stockage afin de rendre contigus des éléments et pouvoir créer des vecteurs;

– utiliser des sous-programmes vectorisés de la bibliothèque MAGEV (Mathematical and Geophysical Vector Library).

Ceci va nous conduire à modifier la structure de l'algorithme lui-même.

Remarquons que les calculs portant uniquement sur le signal complexe (multiplication par une constante, affectation, ...) se vectorisent automatiquement, mais cela concerne un nombre peu important de boucles.

Nous nous sommes également intéressés à la vectorisation du calcul du recadrage linéaire et du signal analytique qui peut être effectuée très rapidement.

4.3. Vectorisation du calcul de la TWVD

4.3.1. *Vectorisation des TFD*

Le calcul des TFD représente la partie prépondérante du calcul de la TWVD. Il sera vectorisé en utilisant un sous-programme vectorisé de calcul de FFT, FFTID, de la bibliothèque MAGEV. Son principe consiste à calculer en parallèle plusieurs FFT. Ceci est tout à fait intéressant pour notre application puisque l'on doit calculer N FFT. Au lieu de calculer séquentiellement une ligne, puis de faire sa FFT, on calculera donc globalement toutes les lignes, puis on utilisera FFT1D. Ceci impose donc de créer un tableau R bi-dimensionnel contenant les différentes lignes à transformer. De plus, les performances de ce sous-programme sont maximales lorsque les données sont « décomplexifiées », c'est-à-dire lorsque les parties réelles et imaginaires sont séparées.

Le stockage requis pour le tableau R est alors représenté sur le schéma I :

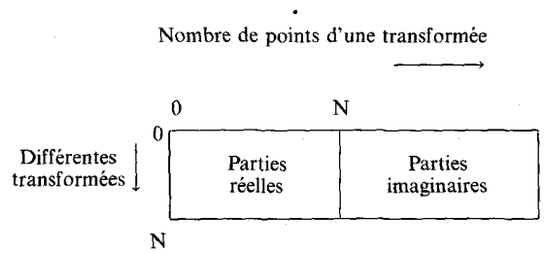


Schéma I

4.3.2. *Calcul du tableau R*

En FORTRAN, le rangement d'un tableau bi-dimensionnel se fait par « colonne », c'est-à-dire que les éléments d'un tableau X, N × N, sont dans l'ordre :

X (1,1), X (2,1) ... X (N, 1) X(1,2), X(2,2) ... X (N, N)

Du fait de la condition de contiguité des éléments d'un vecteur sur le Cyber 205, il sera donc plus intéressant de créer le tableau R par colonnes plutôt que par lignes. Pour cela, on regroupera les lignes paires (resp. impaires) dans la partie supérieure (resp. inférieure) du tableau. Les calculs vectoriels seront effectués verticalement sur des vecteurs de longueur N/2 puisque les traitements sur les lignes paires et impaires sont différents. Ceci revient à permuter l'ordre des boucles qui définissaient les tableaux $\{r_k/k=0, N-1\}$ en :

```
Pour k=0, N-1 faire
  Pour p=0, N/2-1 faire
    R (p, k)=z_{p+k} z*_{p-k}
  fin pour
fin pour
```

pour la première moitié du tableau et :

```
Pour k=0, N-1 faire
  Pour p=0, N/2-1 faire
    R(p+N/2, k)=z_{p+k} z*_{p-k+1}
  fin pour
fin pour
```

pour la seconde moitié du tableau.

A cause de la symétrie des éléments de R (exposée au paragraphe 2.4), les calculs ne seront faits que pour k variant de 0 à N/2, l'autre moitié s'obtenant par conjugaison.

Toutefois, nous avons vu qu'afin d'avoir un gain maximal dans FFT1D, il convient de décomplexifier le tableau à traiter. En combinant cette exigence avec le calcul vectoriel des lignes paires et impaires, il convient de créer un tableau N × 2N dont la structure est représentée sur le schéma II.

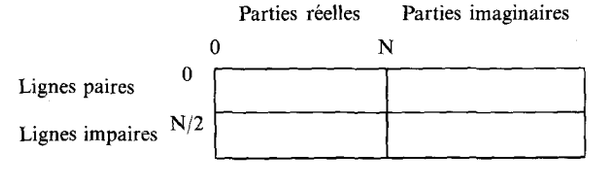


Schéma II

Ceci pourrait être fait au moment du stockage des éléments dans le tableau R, après avoir fait les calculs en arithmétique complexe. Toutefois, ici il est plus économique d'effectuer les calculs en arithmétique réelle simulant l'arithmétique complexe. En effet, on gagne ainsi du temps pour chaque opération complexe, à la fois, sur la séparation en parties réelle et imaginaire, sur le calcul effectif, et sur la fusion des résultats des parties réelle et imaginaire dans le vecteur

complexe résultat, ce qui serait ici fait en pure perte puisqu'il faudrait alors reséparer ces parties pour les stocker dans le tableau R. On a donc créé une routine vectorisée permettant de simuler une multiplication complexe, adaptée à ce stockage, puisque c'est l'opération de base ici. De plus, la conjugaison complexe devient alors la simple combinaison d'une affectation et d'un changement de signe.

4.4. Vectorisation du recadrage linéaire

Cette partie consiste à calculer le minimum et le maximum de la partie réelle du tableau R (puisque la TWVD est réelle), puis à effectuer le recadrage linéaire. Cette dernière partie est simplement vectorisée en permutant l'ordre des boucles si besoin est. La vectorisation de la recherche du minimum-maximum est équivalente à celle d'un test. Elle est réalisée en calculant le maximum et le minimum de chaque colonne de la partie réelle du tableau R à l'aide des instructions vectorielles spécialisées à cette tâche, Q8SMAX et Q8SMIN. Les minimum et maximum globaux du tableau R sont alors obtenus par l'application de ces deux mêmes instructions aux résultats précédents.

4.5. Vectorisation du calcul du signal analytique

La seule modification apportée à cette partie concerne l'utilisation de FFT1D pour le calcul des FFT, les autres boucles étant vectorisées automatiquement.

5. Résultats

5.1. Performances

Le code vectorisé a été exécuté pour différentes valeurs de N. Les tableaux III et IV comportent respectivement les temps d'exécution et les performances des différentes parties du programme. Les temps sont exprimés en millisecondes et les performances en mégaflops.

Tableau III

Temps d'exécution (ms) des trois parties du programme en utilisant l'algorithme vectorisé (calculs sur 64 bits).

N.	64	128	256	512	1 024
TSIG	0,8	0,9	1,1	1,4	1,8
TWV	5,8	12,3	32,8	111,7	424,2
TREC.	0,6	1,7	5,7	18,5	64,2

On constate qu'on obtient de très bonnes performances pour le calcul de la TWVD surtout lorsque la taille du signal d'entrée dépasse 256.

Tableau IV

Vitesse d'exécution (Mflops) des trois parties du programme en utilisant l'algorithme vectorisé (calculs sur 64 bits).

N.	64	128	256	512	1 024
MSIG.	2,1	4,7	8,6	15,9	26,1
MWV.	39,0	79,4	128,4	162,3	183,0
MREC	40,9	57,8	68,9	85,0	97,8

Le gain par rapport à la version scalaire est porté dans le tableau V.

Tableau V

Gain apporté par la vectorisation pour chaque partie du programme (calculs sur 64 bits).

N	64	128	256	512	1 024
Gain (SIG). . . .	1,5	2,7	4,6	7,5	12,1
Gain (WVD). . .	16,2	31,3	48,5	58,8	63,5
Gain (REC) . . .	16,0	22,4	26,7	33,5	38,6

Ces résultats ont été obtenus en utilisant l'arithmétique conventionnelle du Cyber 205 (64 bits). En fait, sans perte d'informations, les calculs auraient pu être effectués sur 32 bits, ce qui correspond ici à une arithmétique demi-précision. Dans ce cas, les temps de calcul et les performances atteintes sont portées dans les tableaux VI et VII.

Tableau VI

Temps d'exécution (ms) des trois parties du programme en utilisant l'algorithme vectorisé (calculs sur 32 bits).

N.	64	128	256	512	1 024
TSIG	0,8	0,9	1,0	1,1	1,4
TWV	5,0	11,1	27,2	82,6	287,5
TREC.	0,4	1,0	3,2	10,0	33,5

Tableau VII

Vitesse d'exécution (Mflops) des trois parties du programme en utilisant l'algorithme vectorisé (calculs sur 32 bits).

N.	64	128	256	512	1 024
MSIG.	2,2	4,8	10,2	19,6	35,6
MWV.	44,4	87,5	154,5	219,3	270,0
MREC	61,4	98,3	122,8	157,2	187,8

5.2. Exemples d'exécution

Nous donnons, à titre d'exemple, la DWVD de deux signaux discrets simulés correspondant à l'échantillonnage de signaux continus dont on peut calculer analytiquement la DWV continue. Les résultats obtenus sont visualisés sous la forme d'images numériques sur la console d'un système de traitement d'images. Sur les images résultantes, l'axe des temps est porté en abscisse et l'axe des fréquences en ordonnées.

Le premier signal traité est le signal rectangulaire.

$$(19)\quad s(t)=\mathrm{rect}_T\left(t-\frac{T}{2}\right)=\begin{cases}1 & \text{si } 0<t<T\\ 0 & \text{sinon}\end{cases}$$

Dans ce cas, on peut montrer que :

$$(20)\quad W_s(t,\nu)=\frac{\sin(2\pi(T-2|t-T/2|)\nu)}{\pi\nu}\,\mathrm{rect}_T\left(t-\frac{T}{2}\right)$$

La figure 4 (*a*) représente l'image obtenue pour $T=1$ s et $N=256$.

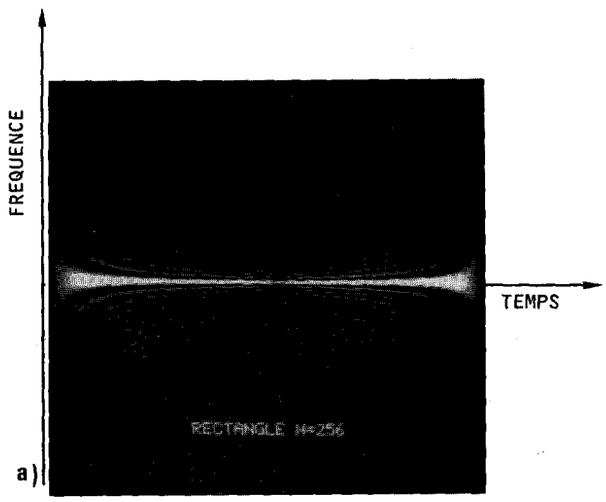
L'exemple suivant porte sur la transformation d'un signal modulé linéairement en fréquence, du type :

$$(21)\quad s(t)=\cos\left(2\pi\left(\nu_0 t+\frac{B}{2T}t^2\right)\right)\mathrm{rect}_T\left(t-\frac{T}{2}\right)$$

La DWV continue du signal analytique qui lui est associé, lorsque $BT\gg 1$, est donnée par :

$$(22)\quad W_s(t,\nu)=\frac{\mathrm{Sin}[2\pi(T-2|t-(T/2)|)(\nu-(Bt/T)-\nu_0)]}{\pi(\nu-(Bt/T)-\nu_0)}\times\mathrm{rect}_T\left(t-\frac{T}{2}\right)$$

Les coupes de cette représentation pour t constant sont des sinus cardinaux. Le lieu des maximums se situe sur la droite $\nu=\nu_0+(Bt/T)$ qui représente la loi de modulation du signal.



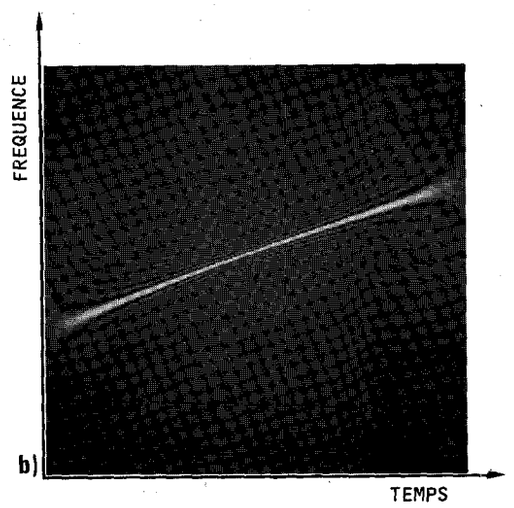


Fig. 4. – Résultats produits par l'exécution du programme dans le cas de deux signaux discrets. La DWVD de ces signaux est obtenue sous la forme d'une image numérique visualisée sur la console d'un système de traitement d'images. Les signaux traités sont : (*a*) un signal rectangulaire; (*b*) un signal modulé linéairement en fréquence.

La TWVD d'un tel signal a été calculée pour les valeurs suivantes des paramètres :

$$\nu_0=35\,\mathrm{Hz},\qquad B=50\,\mathrm{Hz}$$
$$T=1\,\mathrm{s},\qquad N=512$$

Sur l'image 512×512 obtenue, représentée sur la figure 4(*b*) les pas d'échantillonnages temporel et fréquentiel sont respectivement de 1,95 ms et 0,5 Hz. Conformément à la théorie, on observe un maximum d'intensité sur la droite $\nu=35+25t$.

6. Conclusion

Après avoir rappelé la définition et les principales propriétés de la TWV, nous avons discuté le choix d'une version discrète, la TWVD, s'en rapprochant au maximum. Le principe du calcul de cette transformation à partir des échantillons du signal d'entrée a été donné.

Après un rappel sur les calculateurs vectoriels, l'algorithme scalaire a été mis en œuvre sur le Cyber 205. Nous présentons les temps de calcul et performances ainsi obtenus.

Nous nous sommes ensuite attachés à vectoriser cet algorithme pour exploiter pleinement le calculateur. Ceci a consisté à respecter la condition de contiguité (permutations de boucles, changement local de stockage, utilisation de sous-programmes de bibliothèque) et à modifier le stockage global et par là l'algorithme lui-même.

D'autre part, la vectorisation explicite a permis l'écriture d'une syntaxe plus concise qui produit un code

très lisible. Les performances maximales obtenues de 183 Mflops en 64 bits (resp. 270 Mflops en 32 bits) sont excellentes par rapport aux performances maximales du Cyber 205 deux pipes qui sont de 200 Mflops en 64 bits (resp. 400 Mflops en 32 bits).

Le programme fournit un moyen de calcul rapide de la TWVD. Le résultat présenté sous la forme d'une image numérique peut alors être manipulé souplement et exploité sur un système de traitement d'images. Nous pensons qu'il peut être un outil efficace pour l'étude de signaux issus de différents domaines : parole, sonar, radar, télécommunication, . . .

Manuscrit reçu le 11 *septembre* 1985.

BIBLIOGRAPHIE

[1] B. Bouachache et P. Flandrin, Wigner-Ville Analysis of time varying signals, *Proc. ICASSP*, Paris, France, May 1982, p. 1329-1331.

[2] D. S. Chan, A Non-Aliased Discrete-Time Wigner Distribution for Time-Frequency Signal Analysis, *Proc. ICASSP*, Paris, France, May 1982, p. 1325-1328.

[3] T. A. C. M. Claasen et W. F. G. Mecklenbrauker, The Wigner Distribution. A tool for time-frequency signal analysis, *Phillips J. Res.*, 35, 1980, part I : p. 217-250, part II : p. 276-300, part III : p. 372-383.

[4] T. A. C. M. Claasen et W. F. G. Mecklenbrauker, On the time-frequency discrimination enerhy distributions : can they look sharper than Heisenberg ?, *Proc. ICASSP*, San Diego, California, 1984, p. 41.B.7.1-41.B.7.4.

[5] T. A. C. M. Claasen et W. F. G. Mecklenbrauker, The Aliasing Problem in Discrete-Time Wigner distributions, *IEEE Trans. on Acoustics, Speech and signal Processing*, ASSP-31, n° 5, october 1983, p. 1067-1072.

[6] T. S. Duprani et R. Chapman, Systolic Processor for computing the Wigner Distribution, *Electronics letters*, 19, n° 13, June 1983, p. 476-477.

[7] P. Flandrin et B. Escudie. Time and frequency representation of finite energy signals : a physical property as a result of an hilbertian condition, *Signal Processing*, 2, 1980, p. 93-100.

[8] P. Flandrin et B. Escudie, An interpretation of the pseudo-Wigner-Ville distribution, *Signal Processing*, 6, n° 1, January 1984, p. 27-36.

[9] P. Flandrin, Some features of time frequency representations of multicomponent signals, *Proc. ICASSP*, mars 1984, San Diego, California, p. 41.B.4.1-41.B.4.4.

[10] P. Flandrin et B. Escudie, Géométrie des fonctions d'ambiguité et des représentations conjointes de ville : l'approche de la théorie des catastrophes, *Proc. GRETSI*, 1-5 juin 1981, Nice, p. 69-74.

[11] R. W. Hockney et C. R. Jesshope, *Parallel Computers*, J. W. Arrowsmiths Ltd Bristol, Great Britain, 1981.

[12] C. P. Janse et A. J. M. Kaiser, Distribution of Loudspeakers : the application of the Wigner Distribution, *Journal of the audioengineering Soc.*, 31, n° 4, April 83.

[13] A. J. M. Janssen, Gabor Representation and Wigner Distribution of signals, *Proc. ICASSP*, mars 1984, San Diego, California, p. 41.B.2.1-41.B.2.4.

[14] F. Peyrin, D. Vray, G. Gimenez et R. Person, Applications of the Wigner Ville Transform to fish echo signals, *Proc. IEEE MELECON* 85, 8-10 octobre 1985, Madrid, Espagne.

[15] F. Peyrin et R. Prost, A unified definition for the Discrete-time, Discrete frequency and Discrete time-frequency Wigner-Distribution, *IEEE ASSP*, 1986 (à paraître).

[16] J. Ville, Théorie et applications de la notion de signal analytique, *Câbles et Transmission*, 1, 1948, *p.* 61-74.

[17] E. Wigner, On the quantum correction for thermodynamic equilibrium, *Phys. Rev.*, 40, 1932, p. 749-759.

[18] Documentation technique Control Data Cyber 203/205, *User Guide*, n° 84002390.