UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE POUR L'OBTENTION
DU DIPLÔME DE MAÎTRE ÈS SCIENCES APPLIQUÉES EN
ÉLECTRONIQUE INDUSTRIELLE

PAR
MOHAMED BEN SLIMA

ÉTUDE D'APPLICATIONS D'UN PROCESSEUR NUMÉRIQUE DE SIGNAUX (DSP) DANS LES SYSTÈMES DE MESURE ÉLECTRONIQUES

SEPTEMBRE 1993

À mes parents, mes soeurs et mes frères

À tous ceux qui me sont trés chers ...

RÉSUMÉ

Un système de mesure électronique, pour les applications en temps réel, devrait assurer l'exactitude des mesures demandée. La vitesse de traitement des micro-ordinateurs augmente constamment, mais, avec les besoins des algorithmes de traitement des données de plus en plus complexes (pour rendre différents procédés contrôlables), les exigences pour le temps de traitement augmentent davantage.

Une des possibilités d'augmenter la vitesse de mesure d'un système de mesure est l'application des processeurs numériques de signaux (ang. Digital Signal Processor -DSP).

L'objectif principal de ce projet de recherche est l'analyse d'applicabilité du DSP dans les systèmes de mesure électroniques, le développement des algorithmes de reconstitution des signaux et leur implantation dans un processeur numérique de signaux, DSP56000 de Motorola, en vue des applications choisies. Ces applications représentent les besoins les plus urgents dans le domaine de reconstitution de mesurandes.

La première application, basée sur un modèle dynamique linéaire du système de mesure, consiste à corriger les erreurs de conversion A/N de plusieurs signaux multiplexés (dus aux délais de multiplexage) en utilisant les filtres d'interpolation de Lagrange. Les résultats expérimentaux confirment l'efficacité du DSP pour la correction de ses erreurs en temps réel.

La deuxième application aborde le problème de reconstitution de signaux spectrométriques. Les résultats expérimentaux montrent que la méthode de déconvolution spectrale avec la régularisation de Tikhonov offre une solution trés efficace pour l'amélioration de la résolution des

mesures spectrométriques. L'utilisation du DSP est totalement indispensable vu la grande quantité de calcul de FFT demandée.

La troisième application consiste en l'étalonnage statique d'un système de mesure électronique pour l'analyse ultrasonore des solutions, basé sur son modèle non linéaire. Les fonctions spline d'interpolation sont utilisées à cette fin. Les résultats expérimentaux montrent l'applicabilité de DSP pour l'étalonnage des systèmes de mesure.

Les applications élaborées, dans ce mémoire, représentent une large classe d'applications possibles du DSP dans le domaine de reconstitution de mesurande. Cela confirme l'applicabilité du DSP, et par extrapolation logique de processeurs spécialisés (dédiés) dans les systèmes de mesure. Les résultats de ce travail constituent une contribution à l'intégration des systèmes de mesure. L'intégration fonctionnelle et technologique représente une tendance actuelle dans le développement du domaine de systèmes de mesure.

REMERCIEMENTS

En guise d'introduction à cette thèse, je tiens à souligner ma gratitude à l'égard de ceux qui m'ont assisté tout au long de ce travail. En ce sens, je remercie sincèrement le Docteur Andrzej Barwicz qui a bien voulu assumer la direction de cette thèse. Je le remercie pour son aide, interventions et ses précieux conseils qui ont facilité la réalisation de ce travail.

Mes remerciements les plus sincères vont au Docteur Roman Z. Morawski pour m'avoir assisté lors de mon séjour à l'Institut Radioélectronique de l'École Polytechnique de Varsovie en Pologne. Son étroite collaboration, son aide précieuse ainsi que ses nombreux conseils, m'ont guidé tout au cours de la recherche et de la rédaction.

Je suis trés heureux que Monsieur Janusz Konrad, professeur à l'I.N.R.S. Télecommunications soit membre du jury. J'en suis flatté et l'en remercie trés sincèrement.

Je voudrais, par l'occasion qui m'est offerte, remercier tous les professeurs de l'École Nationale des Ingénieurs de Sfax et de l'Université du Québec à Trois-Rivières pour leurs dévouements, compréhensions et efforts mis à ma disposition pendant tant le temps de mes études universitaires.

J'exprime mes remerciements à tous les membres du personnel de la Direction de la Coopération Internationale et en particulier Madame Yveline Coté, pour l'occasion qu'ils m'ont offerte pour étudier au Canada et pour leur appui financier.

Je tiens aussi à remercier tous les membres du personnel du Ministère de l'Enseignement Supérieur de la Tunisie pour l'occasion qu'ils m'ont donnée pour poursuivre mes études supérieures.

Je voudrais, finalement, adresser mes remerciements à tous les collègues pour le soutien et l'encouragement qu'ils m'ont apportés et pour leur amitié. Je voudrais mentionner en particulier Mnif Faiçal, Slim Souheil, Kaffel Maher et Ennaceur Mohamed. J'exprime à tous ma profonde gratitude.

TABLES DES MATIERES

CHAPITRE II
MODÉLISATION MATHÉMATIQUE DU PROBLEME DE RECONSTITUTION DU MESURANDE

CHAPITRE III
APPLICATION DU PROCESSEUR NUMÉRIQUE DE SIGNAUX POUR LA RECONSTITUTION DU MESURANDE

CHAPITRE IV

INTERPRÉTATION DES RESULTATS DES APPLICATIONS

LISTE DES FIGURES

Figures

INTRODUCTION

Le signal de mesure représente l'information sur le phénomène physique étudié. Il peut être masqué par des perturbations indésirables. Le traitement de signal a pour but d'extraire de ce signal l'information utile à l'utilisateur.

En vue d'extraire l'information des signaux mesurés, un système de mesure électronique, aprés avoir converti les signaux mesurés en signaux électriques (conversion A/A), procéder à la conversion A/N suivie par le traitement des données (conversion N/N des résultas de mesure partielles) dans le domaine des signaux électriques. La technique de traitement utilisée et la vitesse de mesure caractérisent principalement un système de mesure.

Historiquement, le traitement analogique a précédé le traitement numérique, appliqué aux signaux; cependant l'utilisation de cette dernière technique a tendance à se généraliser.

En réalité, les techniques de traitement numérique de signal sont plus anciennes, mais n'avaient donné lieu qu'a trés peu de réalisations concrètes par rapport aux solutions analogiques. En effet, leur mise en oeuvre autour de circuits de traitement de base comme les additionneurs, les multiplicateurs et autres registres à décalage, aboutissaient à des solutions performantes mais encombrantes et gourmandes en énergie. Les microprocesseurs quant à eux n'étaient pas tout à fait adaptés à ces applications où les exigences de rapidité de traitement et d'exactitude étaient trop importantes. Baptisés DSP (pour

Digital Signal Processor) les processeurs numériques de signaux viennent pour répondre aux besoins en performances nécessités par le traitement numérique des signaux en temp réel.

Mais, ces processeurs ont jusqu'a présent, surtout, été employés dans les domaines de télécommunication et de traitement de la parole ou presque comme coprocesseurs. Ils sont rarement appliqués aux domaines de l'instrumentation et du contrôle de processus malgré leur rapidité de calcul, utilisant des unités arithmétiques à virgule fixe ou à virgule flottante.

Parmi les problèmes rencontrés dans le domaine de l'instrumentation et de mesure, la reconstitution des mesurandes : il s'agit de retrouver le signal original à partir de sa mesure avec un nombre de points limité. Les applications de la reconstitution sont importantes soit en communication, en géophysique, soit pour les applications qui demandent une haute résolution de la mesure des mesurandes.

L'objectif général de la mémoire est l'analyse des besoins d'applications du DSP dans les systèmes de mesure électroniques et le développement des algorithmes de reconstitution des signaux ainsi que le traitement des données, utilisant un processeur numérique de signaux, DSP56000 de Motorola, pour les applications choisies.

Ces applications sont choisies suivant les besoins les plus souvent dans le domaine de reconstitution de mesurandes, en particulier :

- la correction en temps réel des erreurs de la conversion A/N (dûes aux délais de multiplexage) de plusieurs signaux multiplexés, en utilisant les filtres développés à partir des polynômes de Lagrange [BELLEMARE'89],
- l'amélioration de la résolution des données spectrométriques en utilisant la méthode de déconvolution spectrale avec la régularisation de Tikhonov [MIEKINA & MORAWSKI '86],

- l'étalonnage statique d'un système de mesure électronique pour l'analyse ultrasonore des solutions, en utilisant les fonctions d'interpolation spline [BARWICZ & al. '90],

sera entreprise.

Il existe un livre [EL-SHARKAWY'90] et articles [SRIDHARAN & DICKMAN '88] qui traitent quelques applications du DSP56000 dans le domaine de télécommunication et du filtrage numérique. Mais, peu d'applications sont conçues dans le domaine de l'instrumentation et de mesure.

Les applications qui vont être abordées, dans cette mémoire, viennent pour élargir le champ d'applications de ce processeur dans le domaine de l'instrumentation et ouvrent la voie à d'autres applications.

Cette mémoire est organisée de la façon suivante :

Àu chapitre I, un rappel sur les systèmes de mesure et les éléments qui les constituent ainsi que les techniques de traitement de signaux de mesure sont présentés.

Àu chapitre II, la modélisation mathématique du problème de reconstitution des mesurandes ainsi que le choix des applications seront exposés.

Àu chapitre III, les applications choisies seront développées et analysées. Des résultats de simulations et d'expérimentations avec le DSP56000 seront présentés.

Àu chapitre IV, une interprétation des résultats des applications ainsi que des conclusions sont tirées.

CHAPITRE I

TRAITEMENT DE SIGNAUX DANS LES SYSTEMES DE MESURE ÉLECTRONIQUES

I-1 Définitions

La mesure est une opération fondamentale de la méthode scientifique. Elle permet de vérifier les lois et théories scientifiques et de les exprimer par des expressions mathématiques concises .

Pour choisir, en fonction de la situation, un procédé de mesure approprié, il est indispensable de posséder une connaissance suffisante des méthodes de la métrologie et de leur mise en oeuvre par les moyens techniques disponibles. En effet, une opération de mesure nécessite, généralement, que l'information qu'elle délivre soit transmise à distance du point où elle est saisie, protégée contre l'altération par des phénomènes parasites, amplifiée, avant d'être traitée et exploitée par différents moyens. Ces fonctions sont assurées par des dispositifs (éléments) qui constituent un système dit " système de mesure ".

La mesure des grandeurs physiques et électriques, ou le dialogue entre l'opérateur et les éléments constituant le système de mesure, se fait généralement à l'aide de signaux (modèles du temps) dont la nature est complexe et peut être masquée par des perturbations indésirables (bruit de fond).

L'extraction des informations utiles incorporées à ces signaux (par

analyse, filtrage, estimation, etc.) et la présentation des résultats sous une forme appropriée à l'utilisateur ou à l'appareil de mesure, constitue l'une des taches essentielles dévolues au traitement de signaux.

En effet, le traitement de signaux dans les systèmes de mesure constitue la base fondamentale de l'interprétation et de l'analyse de la nature des altérations ou modifications subies par les signaux lors de leur passage au travers des blocs fonctionnels, des dispositifs généralement électriques ou électroniques, constituant le système de mesure. En plus, il fournit, graçe à des méthodes mathématiques, les moyens d'interprétations et de mise en évidence des signaux.

I-2 Systèmes de mesure électriques

I-2-1 Structure d'un système de mesure électrique

La modélisation des systèmes de mesure, [PARATTE'86], a pour but de donner une réponse à la question fondamentale: de quelle façon la représentation d'une grandeur fournie par un instrument ressemble-t-elle à cette grandeur elle même ?. Pour répondre à cette question, les systèmes de mesure sont en général décomposés en un certain nombres d'éléments. Les performances de chaque élément sont décrites au moyen d'un modèle physique adéquat et mises sous la forme d'une caractéristique de transfert.

En effet, la chaîne de mesure est constituée de l'ensemble des éléments, y compris le capteur, rendant possible, dans des conditions données, la détermination précise de la valeur du mesurande [ASCH'87].

À l'entrée de la chaîne, le capteur soumis à l'action du mesurande permet, directement, s'il est actif, ou par le moyen de son conditionneur, s'il est passif, d'injecter dans la chaîne le signal électrique, support de

l'information liée au mesurande. À la sortie de la chaîne, le signal électrique, qu'elle a traitée, est converti sous une forme qui rend possible la lecture directe de la valeur cherchée du mesurande:

- déviation d'un appareil à cadre mobile;
- enregistrement analogique, graphique ou oscillographique;
- affichage ou impression d'un nombre.

Sous sa forme la plus simple, la chaîne de mesure peut donc se réduire à un capteur et à son conditionneur eventuel, associé à un appareil indicateur (fig.1.1)

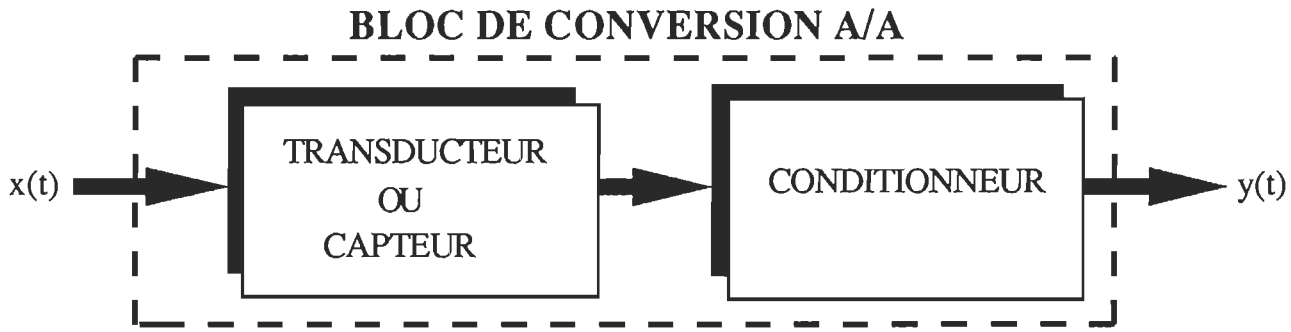


Fig.1.1 Structure simplifiée d'un système de mesure électrique

x (t) : grandeur à mesurer

y (t) : résultat de mesure

Cependant les conditions pratiques de mesure, telles qu'elles sont imposées par l'environnement d'une part, et par les performances exigées pour une exploitation satisfaisante du signal d'autre part, amènent à introduire dans la chaîne des blocs fonctionnels destinés à optimiser l'acquisition et le traitement du signal:

- circuit de linéarisation du signal délivré par le capteur,
- amplificateur d'instrumentation ou d'isolement, destiné à réduire les tensions parasites du mode commun,
- multiplexeur, échantilloneur bloqueur et convertisseur analogique-numérique quand l'information doit être traitée par un processeur.

Pour le cas des mesures électriques des différentes grandeurs

physiques, le signal de mesure est converti en signal électrique. Ce dernier sera traité, afin d'avoir l'information de mesure, selon la structure montrée sur la figure1.2 [BARWICZ '85].

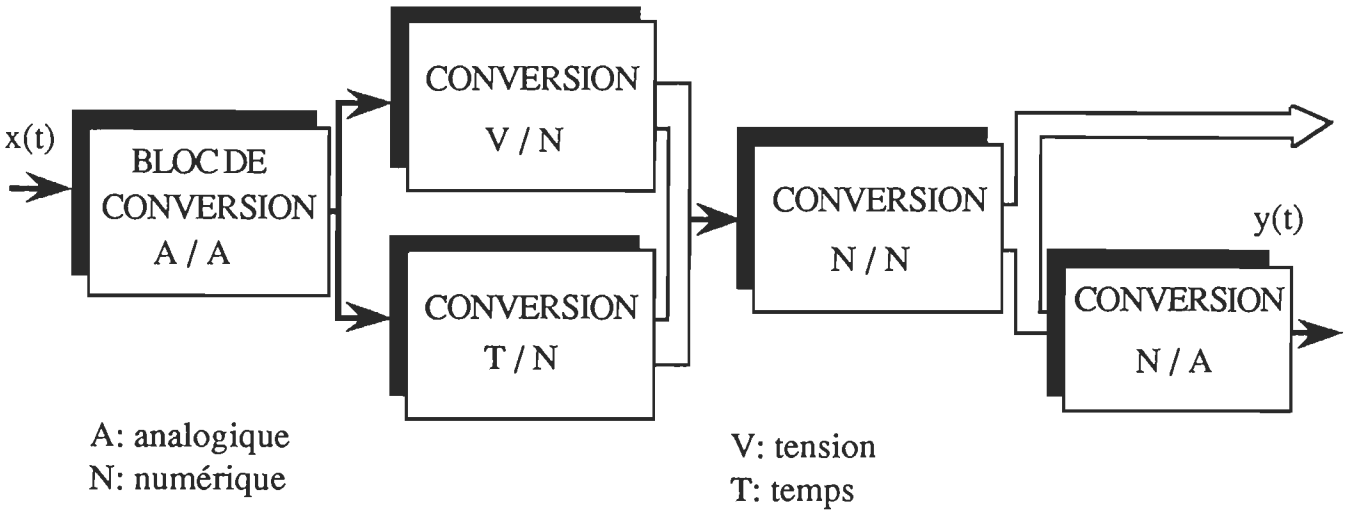


Fig.1.2 Modèle d'un système de mesure électrique

I-2-2 Eléments d'un système de mesure électrique

Vu sa structure, le système de mesure électrique est décrit sous une forme décomposée dans laquelle chaque opération de base apparait de manière explicite. Chaque élément décrit une unité de transformation caractérisée par une grandeur de sortie dépendant d'une grandeur d'entrée ou d'une combinaison de grandeurs d'entrées.

Toutefois, on peut classifier les éléments fonctionnels d'un système de mesure, en se basant sur la constitution d'une structure de mesure électrique (figure 1.2) de la façon suivante [BARWICZ '85]:

- conversion analogique-analogique (A/A) des grandeurs de différentes natures physiques en grandeurs électriques,
- conversion analogique-numérique (A/N) des grandeurs convertibles en tension,

- conversion analogique-numérique (A/N) des grandeurs convertibles en intervalles de temps ,
- conversion numérique-numérique (N/N) ou traitement numérique des données de mesure.

I-2-2-1 Conversion A/A

C'est le premier élément de la chaîne de mesure. Sa fonction est assurée par un capteur ou transducteur approprié, qui traduit quasi-instantanément la grandeur à mesurer (mesurande) en un signal interprétable, par l'intermédiaire d'un système électronique, suivant une loi connue, de toute quantité, propriété ou condition physique que l'on désire déterminer. Les grandeurs mesurées sont couramment une température, un déplacement, une vitesse, une pression, un débit,etc. C'est cette mesure et son interprétation qui renseignent objectivement sur le phénomène ou le procédé. C'est donc la qualité de ce premier élément que dépend l'exactitude ultime de la chaîne de mesure.

I-2-2-2 Conversion A/N et N/A de l'amplitude

L'agencement d'un système de mesure autour d'un ordinateur dépend largement des interfaces dont celui-ci est doté au départ. L'utilisation d'un ordinateur requiert toujours une conversion préalable de tous les signaux analogiques à traiter en signaux digitaux (conversion A/N des tensions électriques dans le cas d'une mesure électrique). Réciproquement les résultats fournis par l'ordinateur sont souvent reconvertis en signaux analogiques lorsqu'il s'agit par exemple de contrôler un processus.

Le convertisseur A/N fait correspondre à une grandeur d'entrée,

continûment variable et de valeur bien définie à chaque instant, une succession de messages numériques. Chacun de ces messages représente la valeur codée de la grandeur d'entrée, c'est le code apparaissant à la sortie du convertisseur.

Le convertisseur N/A, proprement dit, fait correspondre à chaque code numérique une amplitude discrète valable à l'instant d'échantillonnage. Généralement, il sera suivi par un circuit d'interpolation ou d'extrapolation capable de restituer une valeur de sortie entre deux instants d'échantillonnages.

La qualité et l'exactitude de traitement des signaux dépendent, essentiellement, des caractéristiques des éléments de conversion.

I-2-2-3 Conversion A/N et N/A de l'intervalle de temps

Le bloc de conversion analogique-numérique du temps, assure la discrétisation de l'échelle du temps en des intervalles égaux. Il constitue la base de la mesure numérique du temps, fréquence et leurs dérives.

De même que la conversion numérique-analogique du temps du discret au continu, elle est utilisée surtout dans la programmation des valeurs standards du temps.

I-2-2-4 Conversion N/N des données

Ce bloc occupe une place importante dans la chaîne de mesure électrique, il assure le traitement numérique des données issues des blocs de conversion A/N. Selon ses capacités de traitement, on peut également lui assigner des taches de filtrage, de compensation des non-linéarités des capteurs, d'analyse statistique de données mesurées, etc.

I-2-3 Caractéristiques métrologiques des systèmes de mesure

Les performances d'un système de mesure électrique sont liées aux caractéristiques métrologiques des éléments et des instruments de mesure qui le constitue. Ces caractéristiques sont nombreuses, nous nous contenterons d'énumérer les principales, elles sont tirées du livre "Vocabulaire international des termes fondamentaux et généraux de métrologie -1987 ".

- Sensibilité : elle est définie comme le quotient de l'accroissement de la réponse par l'accroissement correspondant du signal d'entrée.
- Mobilité : c'est l'aptitude d'un instrument de mesure de réagir aux petites variations du signal d'entrée.
- Finesse(discrétion) :elle caractérise l'aptitude d'un instrument à ne pas modifier la valeur de la grandeur mesurée.
- Justesse : c'est l'aptitude d'un instrument à donner des indications qui, en moyenne, correspondent à la valeur vraie ou à la valeur conventionnellement vraie de la grandeur meusrée.
- Fidélité : c'est l'aptitude d'un instrument à donner la même indication pour une même valeur de la grandeur mesurée.
- Rapidité : un système est dit rapide s'il est capable de suivre l'évolution de la grandeur d'entrée.
- Exactitude des mesures

Ces caractéristiques ont une influence énorme sur les performances d'un système de mesure électrique.

I-3 Caractérisation des techniques de traitement de signaux

Un signal peut se présenter sous différentes formes, selon que son amplitude est une variable continue ou discrète et que la variable temps

est elle même continue ou discrète.

Généralement, les signaux des systèmes de mesure doivent être traités, soit pour extraire de l'information, soit pour les rendre porteurs d'information. Ces traitements sont effectués à l'aide de blocs fonctionnels que l'on appelle blocs de traitement de signaux . Un tel bloc agit sur un signal d'entrée et produit, à sa sortie, un signal qui est sous une forme plus appropriée pour l'utilisation envisagée.

On peut classer les sytèmes de traitement de la même manière que les signaux [DECOULON '84]:

- les systèmes de traitement analogiques qui opèrent sur des signaux analogiques et produisent des signaux analogiques .
- les systèmes de traitement échantillonnés qui agissent sur des signaux échantillonnés et produisent des signaux échantillonnés.
- les systèmes de traitement numériques qui opèrent sur des signaux numériques et produisent des signaux numériques .

I-3-1 Techniques de traitement des signaux analogiques

Le système de traitement analogique est un dispostif, ou un ensemble de dispositifs, qui effectue sur un signal analogique, provenant de l'extérieur, un ensemble d'opérations de base en utilisant des circuits électroniques.

Dans un système de traitement analogique on peut rencontrer les dispositifs électroniques suivants :

- les amplificateurs,
- les multiplicateurs,
- les filtres analogiques,
- les modulateurs et les démodulateurs.

Les amplificateurs jouent un rôle essentiel dans tous les systèmes de mesure électroniques. La plupart des configurations utilisées, avec les amplificateurs, font intervenir le retour en amont d'une fraction de signal de sortie. Les amplificateurs sont conçus de façon à pouvoir remplir certaines fonctions tels que:

- amplifier les signaux différentiels,
- la mise en forme des signaux,
- assurer l'adaptation d'impédance entre les étages,
- fixation de la bande passante.

Les multiplicateurs assurent, bien entendu, l'opération de multiplication du signal d'entrée avec un signal auxiliaire.

Le filtrage constitue une opération fondamentale dans les techniques de traitement de signaux. Il consiste à séparer les composantes du signal selon leurs fréquences et d'extraire le signal utile, en éliminant les signaux parasites ou accessoires. Le filtre est le circuit qui réalise ces opérations. On distingue deux types de filtres analogiques :

- filtres analogiques passifs composés d'inductances et de capacités,
- filtres analogiques actifs composés d'amplificateurs opérationnels de capacités et de résistances.

Un modulateur est un dispositif paramètrique, linéaire ou non-linéaire, qui produit un signal de sortie dont un ou plusieurs paramètres varient en fonction du signal d'entrée. Il est parfois utilisé dans le système de mesure pour réaliser des transpositions de fréquence façilitant le traitement du signal.

Les démodulateurs assurent l'opération inverse des modulateurs. C'est la reconstitution du signal modulant à partir du signal modulé.

Un exemple de chaîne de traitement analogique de signaux est présenté sur le schéma suivante (figure 1.3)

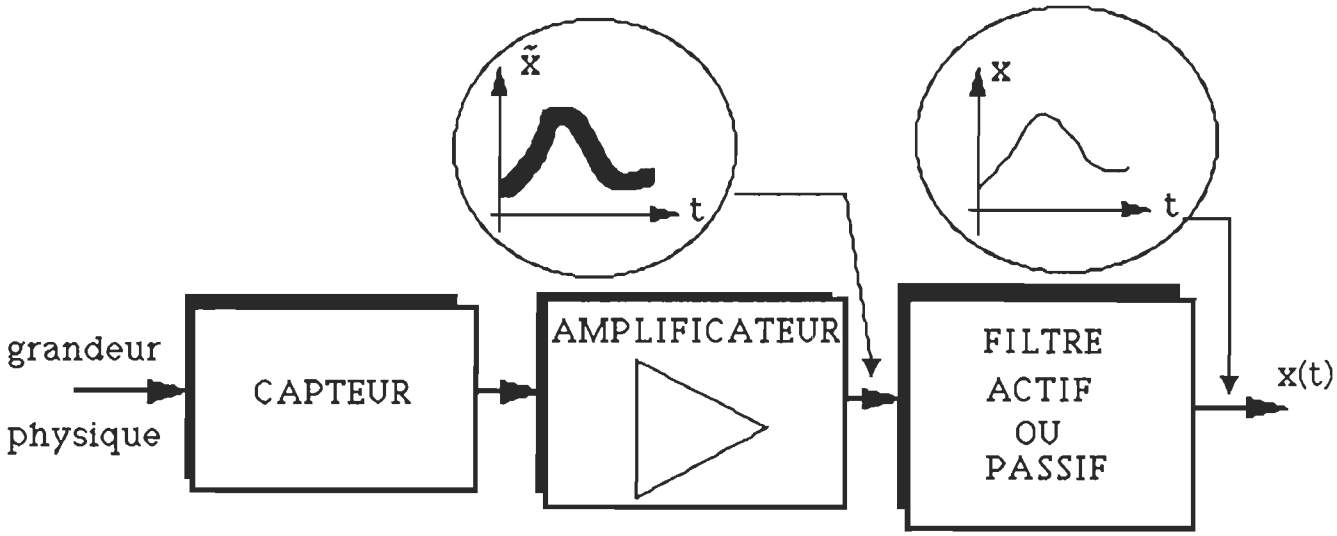


Fig.1.3 Exemple de chaîne de traitement analogique de signaux

I-3-2 Techniques de traitement des signaux échantillonnés

Un signal échantillonné est un signal à amplitude continue et temps discret. Pour réaliser l'opération d'échantillonnage, il faut disposer d'un échantillonneur pour rendre le signal sous forme appropriée à l'utilisation pour le traitement échantillonné. Les dispositifs électroniques utilisés sont :

- les circuits à transfert de charges,
- les filtres à capacités commutées : un tripôle composé de capacités, d'interrupteurs périodiques et d'amplificateurs opérationnels.

I-3-3 Techniques de traitement de signaux numériques

Le traitement numérique du signal porte directement sur des signaux numériques. Il consiste en des séquences d'opérations sur des nombres (calculs arithmétiques, corrélation, transformée de Fourier, comparaisons des nombres, etc.) obeissant à des algorithmes appropriés.

Ce traitement est, généralement, assuré par les processeurs numériques.

Néanmoins, le traitement numérique du signal est appliqué aussi dans le traitement de signaux analogiques selon la chaîne montrée sur la figure 1.4 .

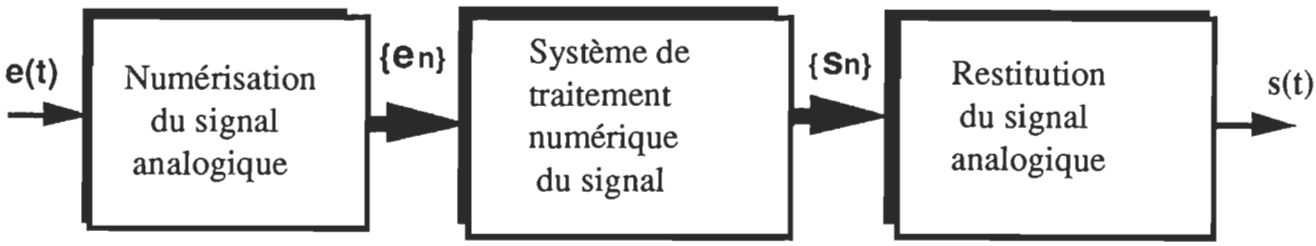


Fig.1.4 chaîne de traitement numérique des signaux analogiques

Le bloc de numérisation du signal assure l'obtention de la séquence discrète { e_n} au moyen de deux opérations :

- l'opération d'échantillonnage (discrétisation dans le domaine de temps) réalisée par un échantillonneur,
- l'opération de quantification (discrétisation dans le domaine d'amplitude) ou de codage réalisée par un quantificateur (convertisseur A/N).

Le bloc de restitution du signal assure l'obtention du signal analogique de sortie s(t) par transformation de la séquence {s_n} obtenue à l'issue du bloc de traitement numérique. Cette transformation est réalisée au moyen d'un convertisseur numérique-analogique suivi d'un interpolateur ou extrapolateur.

I-3-4 Moyens mathématiques de traitement de signaux

I-3-4-1 Classification des méthodes de traitement de signaux

Les outils mathématiques, employés dans le domaine des techniques de traitement de signaux, sont répartis en des groupes selon le type du signal à traiter, déterministes ou aléatoires, et le domaine de traitement

(temporel ou fréquentiel). En effet, on a deux types fondamentaux de signaux [KUNT'80], [DECOULON '84] :

- les signaux déterministes, dont l'évolution en fonction du temps peut être sans ambiguité prédite par un modèle mathématique approprié,
- les signaux aléatoires, dont le comportement temporel est imprévisible et pour la description desquels il faut se contenter d'observations statistiques.

Les méthodes mathématiques utilisées pour le traitement de signaux déterministes sont :

- la transformée de Fourier,
- le filtrage (convolution, corrélation),

Les signaux aléatoires dépendent d'une certaine manière des lois du hasard. Ils ne possèdent pas des représentations temporelles uniques. La description statistique de ces signaux, est généralement obtenue à partir des lois de probabilité des variables aléatoires qui représentent les valeurs de ces signaux. C'est pourquoi les méthodes classiques de traitement de signaux aléatoires utilisent des méthodes probabilistiques, basées sur la théorie de probabilité, tel que les méthodes d'estimation et de décision.

I-3-4-2 Caractérisation des méthodes classiques

Transformée en Z :

La transformée en Z, X(z), d'un signal causal x(t) est donnée par :

$$X(z) = \sum_{n=0}^{\infty} x_n \cdot z^{-n} \qquad (1\text{-}1)$$

où $x_n = x(nT)$, T est la période d'échantillonnage et n un entier.

Cet outil mathématique est trés important lors de la conception et le développement des systèmes de traitement numériques; il permet, par exemple, de représenter un signal possédant une infinité d'échantillons par un ensemble fini de nombres. Ces nombres, caractérisant complètement le signal, permettent de le reconstituer entièrement.

Transformée de Fourier :

Soit la séquence $\{x_n\}$ représentant la séquence des échantillons d'un signal prélevée avec la période T. La transformée de Fourier de cette séquence est donnée par la relation :

$$X(f) = \sum_{n=0}^{\infty} x_n \cdot e^{-j2\pi nfT} \qquad (1-2)$$

On peut remarquer aussi qu'a partir de la transformée en Z, en remplaçant z par le terme $e^{-j2\pi fT}$, on coincide avec la transformée de Fourier du signal. Si la séquence $\{x_n\}$ est finie (jusqu'a N), on défini alors la transformée de Fourier discrète par la relation :

$$X(k) = \sum_{n=0}^{N-1} x_n \cdot e^{-j2\pi n \frac{k}{N}} \qquad (1-3)$$

La transformée de Fourier discrète inverse est donnée par :

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j2\pi n \frac{k}{N}} \qquad (1-4)$$

Cet outil mathématique est trés utilisée quand on analyse dans le domaine de fréquence (analyse spectrale).

Filtrage numérique

On distingue deux grands groupes de filtrages :

- le filtrage numérique temporel,
- le filtrage numérique fréquentiel.

Le filtrage numérique temporel est défini comme étant l'opération d'interruption ou d'atténuation d'un signal. Si on a la séquence $\{x_n\}$ à filtrer, et la séquence $\{g_n\}$ qui représente le filtre alors, filtrer $\{x_n\}$ par $\{g_n\}$ c'est réaliser tout simplement le produit des valeurs des deux séquences au même instant d'échantillonnage. Toutefois, ce filtrage temporel affecte le spectre en fréquence du signal filtré par convolution car on a d'aprés le théorème de Plancherel [MAX'85]:

$$\{ x_n \cdot g_n \} \quad <====> \quad X(j\omega) * G(j\omega) \qquad (1\text{-}5)$$

où $*$: représente l'opération de convolution, $X(j\omega)$ et $G(j\omega)$ - transformée de Fourier discrète des séquences $\{x_n\}$ et $\{g_n\}$ respectivement.

Le filtrage numérique fréquentiel fait appel dans ce cas au transformée de Fourier discrète des deux séquences $\{x_n\}$ et $\{g_n\}$. Filtrer $X(j\omega)$ par $G(j\omega)$ c'est aussi réaliser le produit des deux séquences $X(j\omega).G(j\omega)$. Mais, généralement, on ne peut pas se disposer, à la sortie d'un système de meusre électrique quelconque, de la représentation fréquentielle $X(j\omega)$. On ne peut pas, donc, faire le produit $X(j\omega).G(j\omega)$, il faut passer au domaine temporel à partir de la relation:

$$X(j\omega) \cdot G(j\omega) \quad <====> \quad \{ x_n * g_n \} \qquad (1\text{-}6)$$

on a donc $$X(j\omega)_{filtré} = X(j\omega) \cdot G(j\omega) \qquad (1\text{-}7)$$

$$x_{k\,(filtré)} = \sum_{n=0}^{\infty} g_n \cdot x_{k-n} \qquad (1\text{-}8)$$

Corrélation

En traitement de signaux, il est souvent nécessaire de comparer deux signaux $\{x_n\}$ et $\{y_n\}$. Une méthode possible est de décaler l'un des signaux par rapport à l'autre et de mesurer leur similitude en fonction du décalage. Cette fonction s'exprime par [KUNT'80] :

$$\varphi^{o}_{xy}(k) = \sum_{n=-\infty}^{+\infty} x_n \quad y_{n+k} \qquad (1\text{-}9)$$

où aumoins l'un des deux signaux supposés réels est à énergie finie. Le signal $\varphi_{xy}(k)$ est appelé fonction d'intercorrélation de $\{x_n\}$ et de $\{y_n\}$. Si ces deux signaux sont identiques, le signal $\varphi_{xy}(k)$ est appelé fonction d'autocorrélation. Ce signal atteint son maximum pour une valeur de k, lorsque la similitude est la plus grande.

I-4 Reconstitution du mesurande

I-4-1 Formulation du problème de reconstitution

La mesure d'une grandeur physique x(t) est, généralement, effectuée par un instrument qui fournit un signal y(t). Dans le cas important des appareils linéaires, stationnaires et causaux, les deux grandeurs sont liées par une integrale de convolution :

$$y(t) = \int_{-\infty}^{+\infty} x(\tau) \cdot g(t-\tau)\, d\tau = x(t) * g(t) \qquad (1\text{-}10)$$

où g(t) caractérise la réponse impulsionnelle du système. Lorsque ce dernière ne peut pas être raisonnablement approchée par une impulsion de Dirac, l'entrée x(t) doit être restaurée à partir de la sortie mesurée. La sortie est généralement perturbé additivement par diverses parasites dites " bruits ". A partir de cette situation l'utilisateur veut reconstituer le signal de départ traduisant la situation observée, c'est le problème de déconvolution. En d'autre terme, le problème se résume à la réponse à la question suivante: si on connait la réponse impulsionnelle d'un système linéaire et le signal de sortie, est-ce qu'on peut reconstituer le signal d'entrée?. Pour répondre a cette question il faut résoudre l'équation de convolution (1-10). Il s'agit d'un problème mal conditionné vu que le résultat de reconstitution, x(t) estimé, est trés sensible aux bruits qui

affectent le signal de mesure. Dans la pratique, les données de mesure sont toujours bruitées et l'on ne peut envisager de solutions qu'a l'aide de contraintes physiques et externes, dites de régularisation, qui sont déterminées par les informations a priori que l'utilisateur possède sur le système ou l'appareil de mesure. L'utilisation de ces contraintes, en minimisant certains critères d'erreur, permet de réduire l'influence du bruit dans la solution finale [BIRAUD'76]. Pour cela, plusieurs méthodes de reconstitution ont été élaborées afin de résoudre ce problème.

I-4-2 Classification des méthodes de reconstitution

Dans ce paragraphe, nous ne tenterons pas une revue exhaustive des différentes méthodes utilisées. On va essayer de donner une idée générale sur quelques méthodes utilisées pour la reconstitution du mesurande [MORAWSKI'89].

Méthode directe de reconstitution

Elle consiste à la solution directe des équations algébriques résultant de la discrétisation de l'équation (1-10) à partir des données de mesure. Notons ici l'importance du choix du pas d'échantillonnage. En effet, il joue le rôle de paramètre de régularisation; sa valeur est choisie afin de minimiser l'erreur de reconstitution. On distingue deux sous groupes :

- méthodes basées sur le modèle intégral de donnée, elles utilisent l'intègration numérique pour la discrétisation,
- méthodes basées sur le modèle différentiel de données; elles utilisent des équations différentielles, équivalent à l'intègrale de convolution (1-10), pour la discrétisation.

Méthode variationnelle de reconstitution

Elle consiste à contraindre l'ensemble des solutions de signaux qui minimisent un critère d'erreur donné définissant la qualité de reconsti-

tution. Cependant, il vaut mieux introduire dans cet critère l'information a priori concernant le bruit externe afin de donner une meilleur estimation du signal reconstitué.

Méthode probabilistique de reconstitution

Elle consiste à contraindre l'ensemble de solutions en faisant certaines d'entre elles plus probable que d'autre. Cette méthode utilise des informations a priori (statistiques) sur les signaux et les bruits qui les affectent.

Méthode paramétrique de reconstitution

Pour cette méthode, on essaie de modéliser le signal x(t) avec une fonction connue et paramètrées. On essaie de déterminer les paramètres " a ", qui sont inconnus, en minimisant l'erreur entre la mesure vraie et la mesure réelle. Cette paramétrisation du signal peut être linéaire ou non linéaire. Notons que la méthode linéaire est plus utilisée, vu que les algorithmes de reconstitution sont plus simple.

Méthode itérative de reconstitution

Elle consiste à la paramétrisation récursive des solutions en utilisant les processus d'itérations, jusqu'a la convergence à la solution optimale. On distingue deux groupes:

- méthodes stationnaires, où les paramètres ne changent pas avec les itérations,
- méthodes non stationnaires où les paramètres changent avec les itérations.

-**Méthode de transformée** :

Elle consiste à la formulation des contraintes en utilisant différents domaines de transformation, par exemple :

- domaine spectral,
- domaine cepstral.

I-4-3 Applications pratiques des méthodes de reconstitution

Chaque classe de méthodes de reconstitution possède son champ d'application spécial. Notons qu'une fois la méthode a été choisie, pour l'appliquer il faut prendre en considération :

- les conditions et les exigences concernant la spécification du problème de mesure à résoudre, à savoir le modèle mathématique reliant l'entrée à la sortie, les informations a priori sur le bruit et le mode et la vitesse de traitement de données demandés,
- les façilités offertes par les outils matériels et logiciels.

En plus, les méthodes de reconstitution doivent être adaptées face aux possibilités matérielles, à savoir la capacité de mémoire de l'ordinateur, le traitement parallèle et les librairies statiques et numériques disponibles. En effet, ces méthodes sont couramment utilisées dans les domaines de géophysique, de radioastronomie, de spectroscopie, communications, traitement du signal et des images etc.[JANSSON'84], [HUNT'84]

I-5 Conclusion

En vue d'extraire l'information des signaux mesurés, un système de mesure électronique, aprés avoir converti les signaux mesurés en signaux électriques (conversion A/A), procéder à la conversion A/N suivie par le traitement de données (conversion N/N), devrait assurer l'exactitude de mesure exigée. En effet, la qualité des techniques de traitement de signaux utilisée et la vitesse de mesure caractérisent principalement un système de mesure. En d'autre terme, un système de mesure électronique devrait assurer l'exactitude des mesures demandée dans le temps demandé.

CHAPITRE II

MODÉLISATION MATHÉMATIQUE DU PROBLEME DE RECONSTITUTION DU MESURANDE

II-1- Modélisation d'un processus de mesure

L'analyse des systèmes, de façon générale, consiste à les décrire par des lois mathématiques permettant de prévoir leur comportement. C'est le rôle du physicien d'établir la relation mathématique qui relie l'entrée à la sortie du système : c'est le problème de modélisation mathématique du système.

Le problème de reconstitution des signaux est un cas spécial du problème de modélisation inverse qui est un incident de l'interprétation d'une mesure. En effet, un processus de mesure peut être décomposé en deux parties (fig. 2.1) [MORAWSKI '89] :

- conversion ;
- reconstitution.

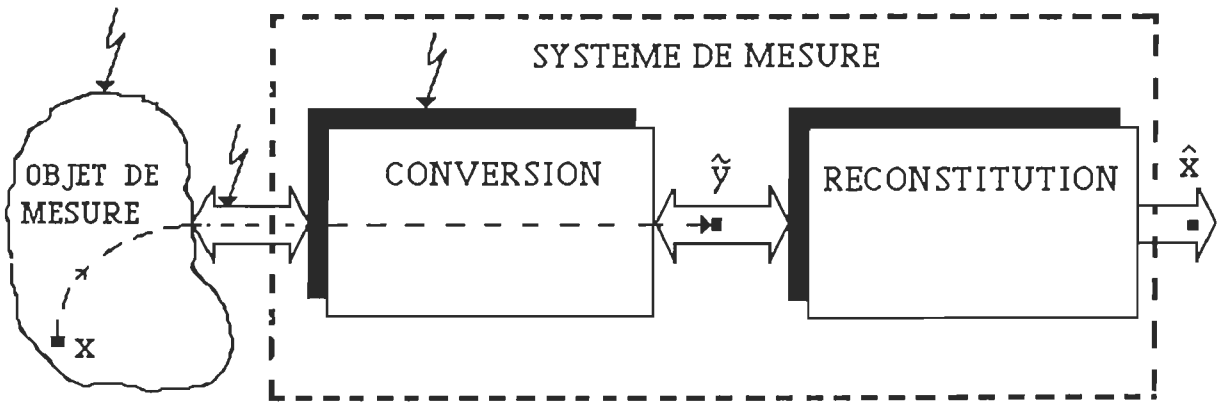


Fig. 2.1 Structure générale d'un processus de mesure

x : grandeur à mesurer (mesurande);

$\tilde{y}$: résultat de conversion ;

$\hat{x}$: résultat de mesure (résultat de reconstitution)

La conversion est une transformation du signal de mesure x (grandeur à mesurer) en un signal y, qui représente x dans un domaine plus approprié à l'utilisateur, tel que un signal électrique ou codes numériques.

La reconstitution consiste à l'estimation du signal de mesure x en se basant sur le résultat de conversion y.

Chaque processus de mesure est perturbé par la présence des grandeurs d'influences, externes et internes, qui affectent non seulement l'acquisition de l'information mais aussi sa transmission.

Pour décrire le problème de reconstitution, on a recourt à un modèle mathématique qui relie le signal y au signal x. Un tel modèle a la forme

$$y = \mathfrak{G}\,[\,x\ ;\ v\,] \qquad (2\text{-}1)$$

où $x = x(t)$ est un vecteur de mesurandes,

$y = y(t)$ est un vecteur de résulats de conversion,

$v = v(t)$ est un vecteur de grandeurs d'influence.

$\mathfrak{G}$ est un opérateur dont son inversion ou pseudoinversion est la clé de la reconstitution des signaux.

D'une façon générale, on peut représenter le modèle mathématique (2-1) par une série de fonctionnelles de Volterra [DE COULON'84 § 8.4]:

$$\mathfrak{G}[\,x(t)\ ;\ v(t)] = \sum_{n=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} g_n(\tau_1, \ldots, \tau_n, v(t)).\ x(t-\tau_1) \ldots x(t-\tau_n)\, d\tau_1 .. d\tau_n \qquad (2\text{-}2)$$

c'est un opérateur homogène de degré n caractérisé par les noyaux $g_n(t_1, \ldots, t_n)$, $n=1,2, \ldots$

II-2 Caractérisation des opérateurs d'identifications

La modélisation mathématique du problème de reconstitution, depend fortement de la nature du comportement de l'opérateur $\mathfrak{g}$. En effet, on peut distinguer deux classes principales :

- opérateurs dynamiques linéaires et non linéaires;
- opérateurs statiques linéaires et non linéaires.

Pour la suite de cette étude, on suppose que dim(v) =0.

Les opérateurs dynamiques linéaires jouissent des propriétés d'additivité, d'homogénéité au cours du temps. Dans le cas de systèmes linéaires, stationnaires et causaux [MAX'85], le signal d'entrée x(t) est lié au signal de sortie y(t) par une intégrale de convolution :

$$y(t) = \int_{-\infty}^{+\infty} x(\tau) \cdot g(t-\tau)\, d\tau \quad = x(t) * g(t) \tag{2-3}$$

donc on aura dans ce cas :

$$\mathfrak{g}\,[\,x(t)\,;\,v(t)\,] = \int_{-\infty}^{\infty} x(\tau) \cdot g(t-\tau)\, d\tau \tag{2-4}$$

En effet, la relation (2-4) découle du modèle général (2-2) dans le cas où $g_n = 0$ pour $n > 1$

L'équation (2-3) est une équation de Fredholm de II^{ième} espèce dont le noyau g ne dépend que de la différence $(t-\tau)$. La signification de g(t) est simple. Si $x(t) = \delta(t)$ (impulsion de Dirac) l'équation (3) sera :

$$y(t) = \delta(t) * g(t) = g(t) \tag{2-5}$$

car la distribution de Dirac est l'unité de convolution, donc g(t) caractérise parfaitement le système linéaire: c'est sa réponse impulsionnelle. Pour la reconstitution du mesurande, dans ce cas, il faut analyser le modèle inverse de l'équation (2-3) en introduisant dans le modèle les

grandeurs d'influences qui affectent les resultats de conversion.

Les opérateurs dynamiques non linéaires forment une vaste classe sans mode de présentation universelle. En effet, aucune théorie globale n'existe qui permette, comme dans le cas linéaire, de déterminer simplement les relations liant la sortie y(t) à l'entrée x(t). Un moyen de relier l'entrée à la sortie, d'un opérateur non linéaire, est la représentation en série de fonctionnelles de Volterra [DE COULON '84 §8.4]:

$$\mathfrak{g}[\, x(t)\, ;\, v(t)] = \sum_{n=1}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} g_n(\tau_1, \ldots, \tau_n, v(t)).\ x(t-\tau_1). \ldots x(t-\tau_n)\, d\tau_1 .. d\tau_n \qquad (2\text{-}6)$$

C'est une extension de la représentation intégrale des opérateurs linéaires (2-3), ce cas correspond au premier terme de la série.

Pour la reconstitution du signal de mesure, pour ce type d'opérateur, le problème réside dans la modélisation des systèmes non linéaires. En effet, il est parfois difficile d'identifier un modèle adéquat du système, ce qui rend la résolution de l'équation (2-6) trés difficile pour reconstituer x(t).

Les opérateurs statiques, quant à eux, ils caractérisent le comportement des systèmes pour des changements de signaux suffisamment lents. Ils forment une classe où la variable temps (t) n'intervienne pas dans les modèles mathématiques.

Les opérateurs statiques linéaires décrivent une dépendance linéaire entre l'entrée et la sortie. Le cas d'une résistance électrique pure, le courant i qui la traverse suit instantanément la tension u appliquée à ses bornes :

$$i = u/R \qquad (2\text{-}7)$$

dans ce cas : $x = u$ et $y = i$. La reconstitution de la grandeur à mesurer est facile pour ce cas.

Les opérateurs statiques non linéaires, quant à eux, décrivent une

relation relativement complexe, par exemple, le cas où le signal de sortie est exprimable en série de puissance de celui de l'entrée:

$$y = \sum_{n=1}^{\infty} \alpha_n \, x^n \qquad (2\text{-}8)$$

Cette relation représente une large gamme d'opérateurs non linéaires suivant les valeurs des coéfficients α_n, n=1,2,. . . . Elle découle du modèle général (2-2) dans le cas où :

$$g_n (\tau_1 , \ldots , \tau_n ; v(t)) = \delta(\tau_1) \ldots \delta(\tau_n) \; \alpha_n(v(t)) \quad \text{pour } n=1,2,\ldots \qquad (2\text{-}9)$$

$$\mathfrak{g}[x(t) ; v(t)] = \sum_{n=1}^{\infty} \alpha_n(v(t)) \; x^n (t) \qquad (2\text{-}10)$$

En conclusion, l'analyse d'un système physique dépend fortement de la nature de l'opérateur $\mathfrak{g}$ (statique ou dynamique). Toutefois, la connaissance du modèle mathématique du système constitue l'étape fondamentale pour la résolution du problème de reconstitution des signaux.

II-3 Choix des applications de la reconstitution du mesurande

Afin de bien analyser ce problème, trois applications seront choisies selon la nature de l'opérateur décrite précédemment. Chaque application traite pratiquement le problème de reconstitution d'une façon différente de l'autre, et envisage des solutions pour la résolution de ce problème.

La première application est basée sur un modèle dynamique linéaire (relation (2-3)) avec un délai idéal. Le modèle mathématique décrit par ce système découle du modèle général (2-2), dans le cas où :

$$g_1 (\tau_1 ; v(t)) = \alpha_1 (v(t)) \, \delta (\tau_1 - \Delta T) \text{ et } g_n = 0 \text{ pour } n > 1 \qquad (2\text{-}11)$$

c'est un opérateur linéaire statique, avec un délai idéal ΔT, de la forme:

$$\mathcal{G}[\, x(t) ; v(t) \,] = \alpha_1(v(t)) \,.\, x (t - \Delta T) \qquad (2\text{-}12)$$

Il s'agit d'un système de mesure destiné aux mesures de plusieurs signaux multiplexés avec un seul convertisseur analogique-numérique. Ces mesures sont affectées par des délais, provoqués par les composants du bloc de conversion, qui introduisent des erreurs sur les valeurs de sorties. Il faut corriger ses erreurs, et reconstruire les signaux d'entrées.

La deuxième application concerne aussi un modèle dynamique linéaire dont la réponse impulsionnelle n'est pas approchée à une impulsion de Dirac. Le modèle mathématique de ce système est donné par la relation (2-4). Il s'agit de la reconstitution de signaux spectrométriques en utilisant la méthode de déconvolution spectrale avec régularisation de Tikhonov.

La troisième application est basée sur un modèle statique non linéaire, décrit par la relation (2-10). Il s'agit de l'étalonnage statique d'un système de mesure électronique pour l'analyse ultrasonore des solutions en utilisant les fonctions spline d'interpolation

Les critères d'évaluation des réalisations pratiques de ces applications sont : l'exactitude de reconstitution du signal de mesure et la vitesse de traitement des données de mesure. Mais, quel sont les outils nécessaires à utiliser pour réaliser ces applications et qui répondent à ces critères?

II-4 Pertinence du processeur numérique de signaux pour les applications

II-4-1 Exigences des blocs de reconstitution du mesurande

Les blocs de reconstitution de signaux font appel aux blocs de traitement numérique de signaux. Ces derniers, font appel aussi aux

techniques et circuits utilisés dans les machines de traitement de l'information. Certains particularités apparaissent cependant, dont la plus marquante est la grande quantité de calcul arithmétiques nécessaire. Il s'en suit que la complexité des blocs dépend étroitement du nombre d'opérations arithmétiques à faire par intervalle de temps élémentaire. En toute sorte, les outils et les blocs de reconstitution de signaux doivent satisfaire les exigences suivantes:

- une exactitude suffisante des résultats de traitement,
- rapidité en réponse et traitement en temps réel,
- Souplesse de point de vue utilisation (programmation, définitions des paramètres, etc.).

En effet la représentation des données, par un nombre de bits fini, doit être suffisamment précise (une bonne plage dynamique). En plus, le temps d'exécution et la rapidité de réponse aux interruptions doivent être capable de suivre l'évolution des données de mesure en temps réel.

Face à ces exigences, on a recourt à des blocs fonctionnels numériques tels que les microprocesseurs, les microcontrôleurs et les processeurs numériques de signaux. Mais lequels des trois blocs peut nous offrir la meilleur reconstitution et la meilleur exactitude de traitement des données de mesure?.

II-4-2 Utilisation des microprocesseurs

Les opérations de traitement numérique(transformée de Fourier, filtrage numérique,etc...) reviennent toutes à exécuter des opérations arithmétiques classiques tels que l'addition et la multiplication. Donc, l'action la plus intérressante à faire pour adapter un tel microprocesseur au traitement numérique consiste à élaborer, surtout, les algorithmes de

multiplication les plus performants en fonction des possibilités du circuit, en particulier le temps d'exécution de l'opération et la plage dynamique du circuit, concernant le nombre de bits de données.

Actuellement, il existe des microprocesseurs de 16, 32 et 64 bits qui peuvent répondre à nos besoins, de point de vu exactitude, mais le problème se pose surtout au niveau de la capacité de traitement en parallèle, de l'existance des registres de travaux et du jeu d'instruction qui n'est pas adapté aux algorithmes de traitement numérique de signaux. De plus leurs architectures n'étaient pas tout à fait adaptées aux applications où les exigences de rapidité de traitement et d'exactitude sont trop importante. Il a fallu, donc, élaborer un nouveau style d'architecture et développer de nouveaux composants avec des vitesses de fonctionnement plus rapide pour satisfaire nos exigences.

II-4-3 Utilisation des processeurs numériques de signaux

Les possibilités offertes par les blocs fonctionnels numériques (microprocesseurs et microcontrôleurs) ont permis d'utiliser des méthodes de traitement de plus en plus complexes, généralement irréalisable par la technique analogique. Ces méthodes deviennent tellement complexes qu'elles exigent, à leur tour, des blocs de plus en plus puissant. Si les microprocesseurs ont une souplesse trés grande pour les traitements, ils présentent cependant une limitation: les traitements complexes ne peuvent s'effectuer instantanément (en temps réel).

Compte tenu des faibles performances des microprocesseurs à usage général en traitement numérique de signal, comparées à celle des structures spécialisées à base de microtranches, et au vu de l'importance du marché potentiel, les fabricants de circuits intégrés ont défini des microprocesseurs spécialisés aux applications du type traitement de

signaux (ang. Digital Signal Processor) en intégrant des architectures éfficients développées à base de microtranches.

Les processeurs numériques de signaux sont basés sur l'architecture Harvard. C'est une structure améliorée de l'architecture de von Neumann, dans laquelle les zones mémoires de données et des instructions possèdent leur propre chemin d'adresse, autorisant ainsi des opérations totalement indépendantes. Cette structure permet une plus grande vitesse de traitement, il n y a pas de goulot d'étranglement sur les bus, un problème souvent rencontré avec l'architecture de von Neumann, où les données et instructions doivent partager un seul et même bus. On retrouve au niveau de ces processeurs des unités de calcul spécialisées incorporant outre une unité arithmétique et logique conventionnelle ou travaillant en arithmétiques adaptées à l'application, un multiplieur parallèle ou un multiplieur-accumulateur ainsi que des unités à décalages et de limitations. Ces organes offrent une grande souplesse de traitement de données et façilitent le traitement en parallèle, ce qui permet une grande vitesse d'exécution des instructions (cycle d'instruction moins que 100 ns) et une haute exactitude des résultats de traitement.

Ces processeurs sont en général classés selon trois grandes groupes selon l'applications envisagées et selon l'orientation du composant :

- processeurs à entrées/sorties spécifiques ;
- processeurs à entrées/sorties à usage général ;
- processeurs spécialisés.

Les processeurs à entrées/sorties spécifiques intègrent l'unité de traitement de données, les mémoires "programme" et "données" ainsi que les interfaces. Ces derniers sont spécialisés, par exemple des convertisseurs A/N et N/A avec multiplexeurs permettant de définir un groupe d'entrées/ sorties analogiques. C'est le cas des processeurs 2920

de INTEL, les TMS32011 et TMS32017 de Texas Instrument (destinés à la téléphonie numérique).

Les processeurs à entrées/ sorties à usage général sont comme l'autre groupe à structure non extensible. Ces composants comprennent donc l'unité de traitement, les mémoires "programme" et "données" ainsi que les interfaces. Ces derniers sont toutefois généraux, c'est-à-dire de type numérique soit série soit parallèle, permettant des connexions aisées avec de larges gammes de composants tels que convertisseurs A/N et N/A (en général selon le mode série) ou autres microprocesseurs et leur famille d'interfaces conventionnels (en général selon le mode parallèle). Ils peuvent être utilisés en processeurs complets pour former des systèmes indépendant ou constituer des coprocesseurs pour des microprocesseurs à usage général. C'est le cas des processeurs 2811 de AMI, µPD7720 de NEC ou HD61810/11 de HITACHI.

Enfin, les processeurs spécialisés qui se caractérisent par de larges possibilités d'extension que ce soit au niveau des mémoires programmes et données ou des interfaces. Ces composants sont généralement prévus pour travailler dans des structures multi-processeurs à partages de resources communes. Ils peuvent travailler seuls ou servir de coprocesseurs à des microprocesseurs à usage général. Cette structure est en général adoptée pour tous les composants modernes et à venir. C'est le cas des processeurs TMS32020/C25/C30 de Texas Instrument, DSP56000 de Motorola, etc.

Le choix entre l'un des trois groupes dépend essentiellement de l'application envisagée et de son cahier de charge. En effet, les applications liées aux traitement de signaux exigent des calculs arithmétiques énorme, ce qui fait que le choix du processeur et des interfaces dont celui-ci est doté, dépend largement du capacité de calcul

et de la rapidité et l'exactitude de traitement. Face à cela, les avantages offerts par le troisième groupe, en ce qui concerne l'extensibilité de la structure et les possibilités de connexions, nous permet de choisir un processeur numérique de signal qui appartient à cette classe pour nos applications (§ II-3): c'est le DSP56000 de Motorola [EL-SHARKAWY '90].

Par opposition à la famille de Texas Instrument TMS320C1x/C2x, ce processeur se caractérise par une structure à haut degré de parallélisme. Il offre en outre des perspectives interésantes vu sa dynamique (mots de 24 bits offrant un bruit d'arrondi faible de 144dB) et un cycle d'instruction moins que 100ns. Il est le premier membre de la famille des composants traitement numérique de signaux, introduite, chez Motorola. Une description du processeur DSP56000/1 se trouve dans l'annexe A-1.

CHAPITRE III

APPLICATION DU PROCESSEUR NUMÉRIQUE DE SIGNAUX POUR LA RECONSTITUTION DU MESURANDE

III-1 CORRECTION DES ERREURS DE MULTIPLEXAGE DANS LA CONVERSION ANALOGIQUE-NUMÉRIQUE DE LA TENSION ÉLECTRIQUE

III-1-1 Formulation du problème

Dans un système de mesure destiné aux mesures de plusieurs signaux avec un seul convertisseur analogique-numérique (A/N) (fig.3.1), la conversion de ces signaux multiplexés ne peut se faire simultanément, car les circuits ne peuvent réaliser qu'une seule opération à la fois. Cela introduit un délai entre le moment où l'on veut faire la mesure d'un signal sur une entrée donnée et celui où, à la sortie du circuit d'acquisition, il est prêt pour sa conversion. Il en résulte que la valeur numérique de sortie ne correspond pas à la valeur analogique du signal d'entrée au moment du début de sa mesure.

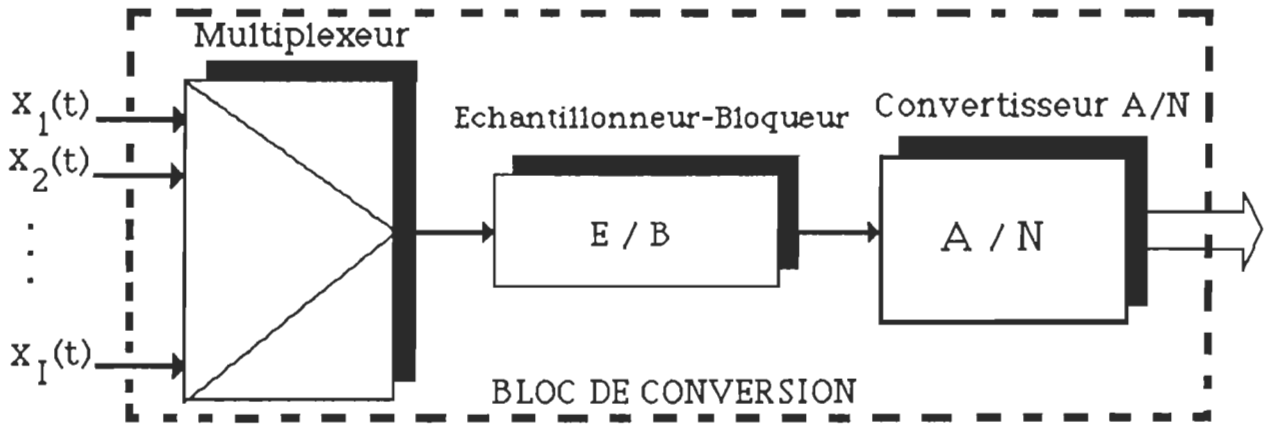


Fig.3.1 Éléments du bloc de conversion A/N de plusieurs signaux

Ce sont les erreurs, introduites par le multiplexeur et l'echantillonneur bloqueur sur le signal à mesurer, qui font que la valeur numérique de sortie n'est pas celle du signal d'entrée au début de son acquisition. Même si ses composants sont électriquement fiables et si les grandeurs d'influences externes sont faibles, la technique d'acquisition, à l'aide de ses composants, introduit des erreurs.

En effet, il existe trois types d'erreurs spécifiques au bloc de conversion analogique-numérique :

- erreur de quantification, reliée à la résolution du convertisseur A/N;
- erreur d'acquisition, reliée au temps que prennent les composants pour réaliser une acquisition;
- erreur de synchronisation vu que l'échantillonneur-bloqueur et le convertisseur A/N sont incapable de traiter plus q'un signal à la fois.

Le délai du bloc de conversion cause une erreur qui dépend fortement du signal à mesurer (son spectre de fréquence). Ce délai est donné par la relation suivante [BELLEMARE '89]:

$$\Delta T_i = (i - 1) \Delta T + T_t \qquad i=1,2,...,I \qquad (3\text{-}1)$$

où ΔT_i : le délai entre l'entrée #1 et l'entrée #i ;

ΔT : le temps entre deux entrées;

T_t : le temps d'acquisition total, c'est le temps entre le début de conversion et le moment ou le signal est prêt à être convertir. Ce temps inclu le délai de multiplexage et le délai de l'échantillonneur-bloqueur.

L'idée de base de l'analyse des erreurs est illustrée à la figure 3.2, dans laquelle , on fait la mesure d'un même signal appliqué sur quatre entrée différentes et chaque entrée est caractérisée par son délai de multiplexage coresspondant ΔT_i indiqué sur la figure 3.3.

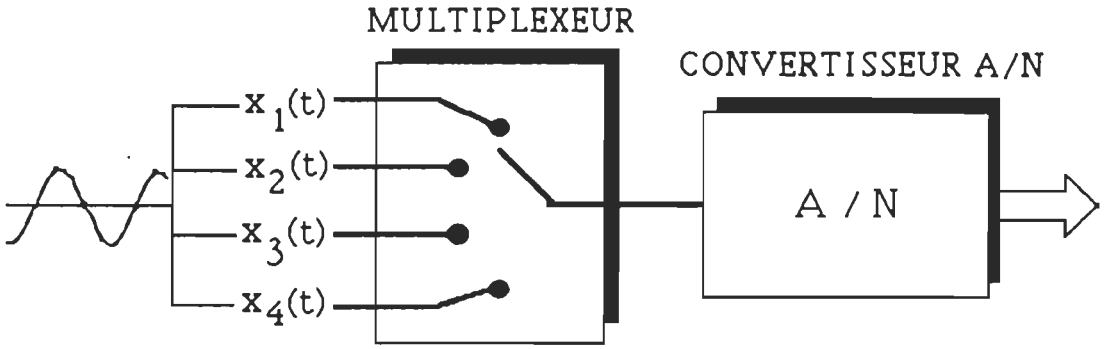


Fig.3.2 Mesure d'un même signal sur quatre entrée différentes

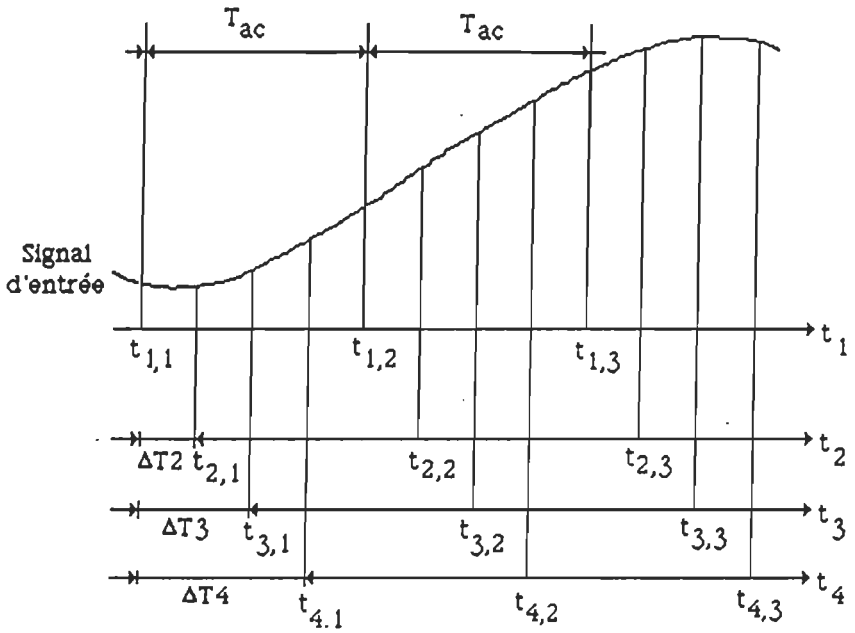


Fig.3.3 Diagramme de temps des quatres signaux d'entrées avec les délais

où t_1, t_2, t_3 et t_4 désigne l'echelle de temps des signaux d'entrée #1, #2, #3 et #4 respectivement et T_{ac} est le temps d'acquisition que prend un seul échantillon pour le même signal d'entrée.

Deux signaux ont été choisis, comme exemple, pour la correction des erreurs de multiplexage :

- un signal sinusoidal de fréquence 1 kHz (fig 3.4)
- un signal sinusoidal de fréquence 100 Hz modulé par un signal

sinusoidal de fréquence 2 kHz. (fig 3.5)

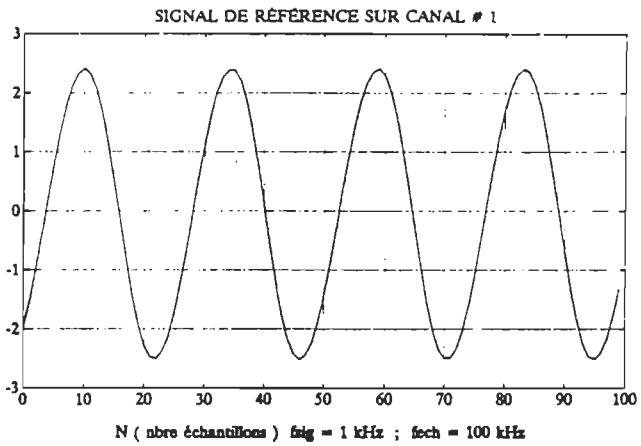


Fig.3.4

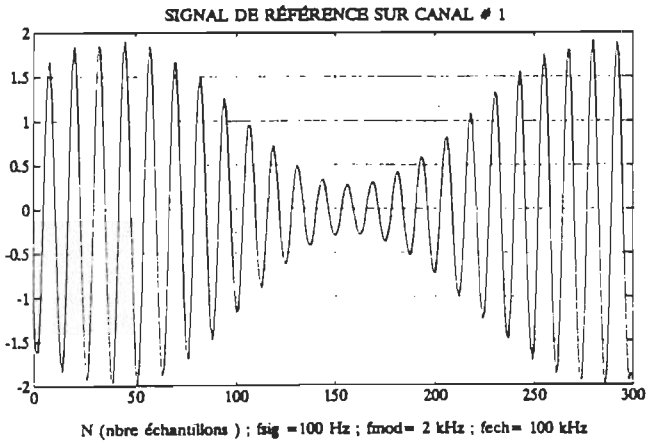


Fig.3.5

Idéalement, les valeurs de sorties devraient être les mêmes au moment de l'acquisition, mais il existe une différence énorme, comme on le voit aux figures (3.4-a), (3.4-b),(3.4-c), (3.5-a), (3.5-b) et (3.5-c), et cela est due aux erreurs de synchronisation.

Le problème consiste donc à diminuer l'influence de ces erreurs causées par les délais en utilisant des méthodes de correction que ce soit dans le domaine fréquentiel ou temporel.

III-1-2 Filtres de correction des délais

Les résultats de l'analyse des délais, par des méthodes de traitement de signaux , démontrent qu'un délai imposé à un signal peut aussi s'écrire comme une modification de la phase du signal [DE COULON'84 § 8.2.17]. La méthode exacte pour corriger ces délais est d'annuler l'erreur de phase directement sur le spectre de phase des signaux, obtenu par transformée de Fourier des signaux mesurés. Ce qui nous amène à étudier la correction dans le domaine fréquentiel.

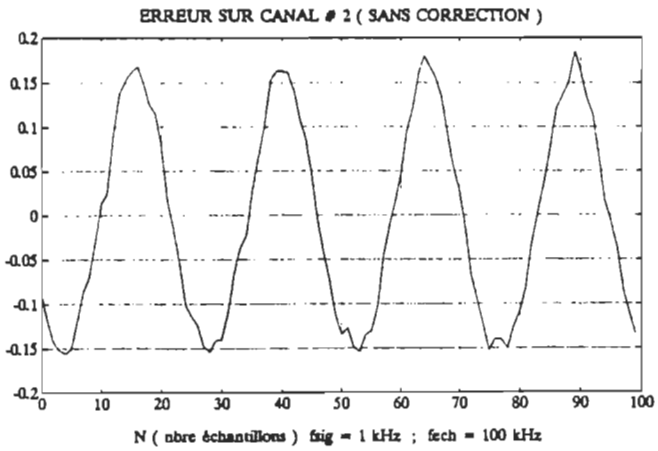


Fig.3.4-a

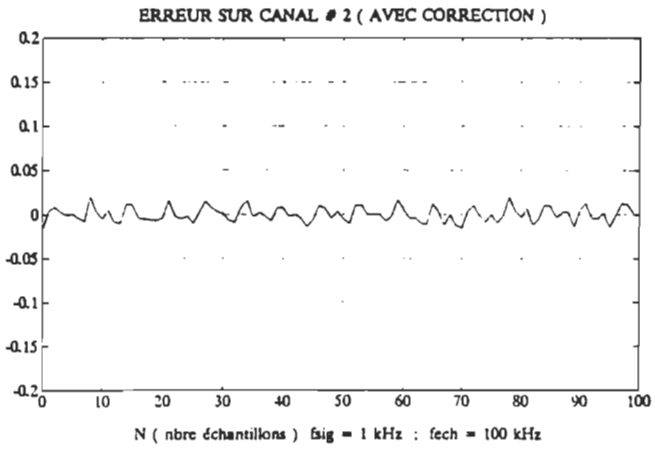


Fig.3.4-d

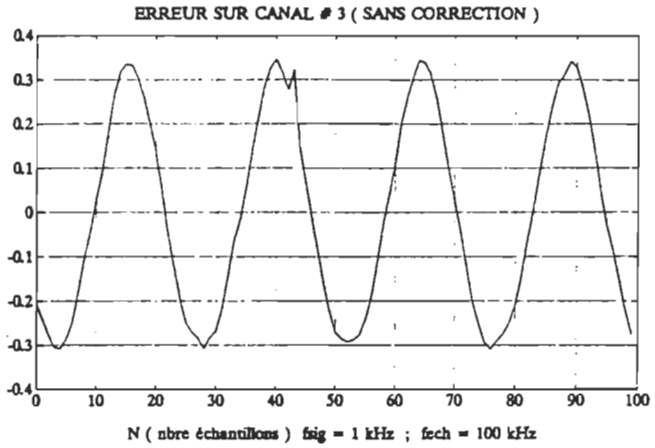


Fig.3.4-b

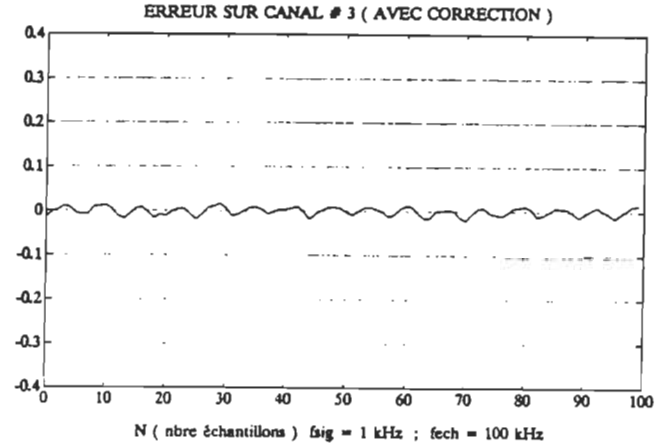


Fig.3.4-e

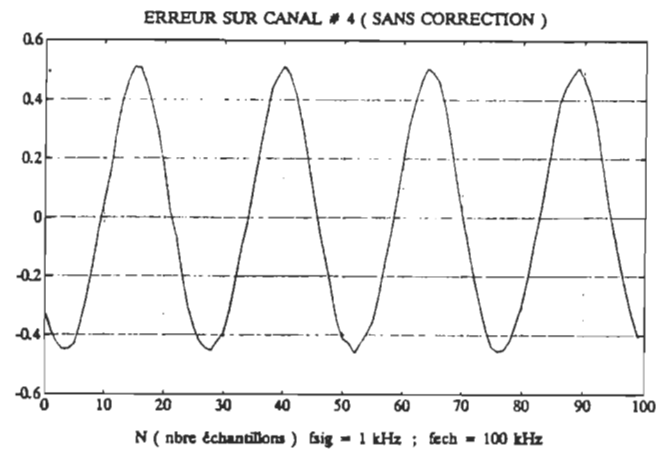


Fig.3.4-c

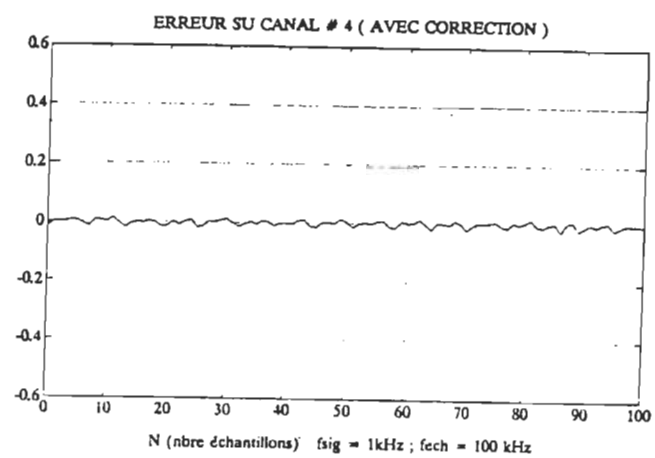


Fig.3.4-f

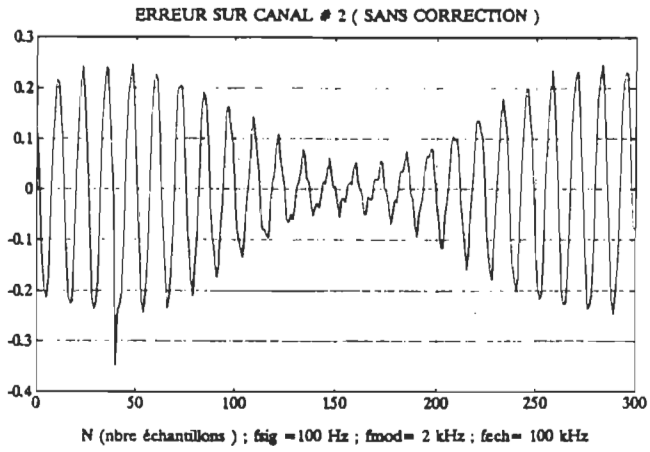


Fig.3.5-a

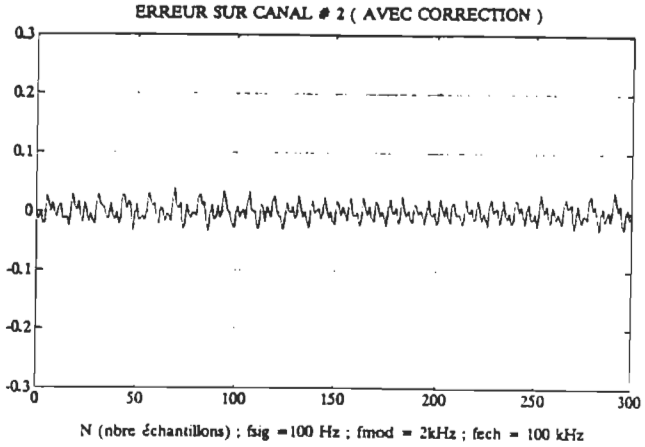


Fig.3.5-d

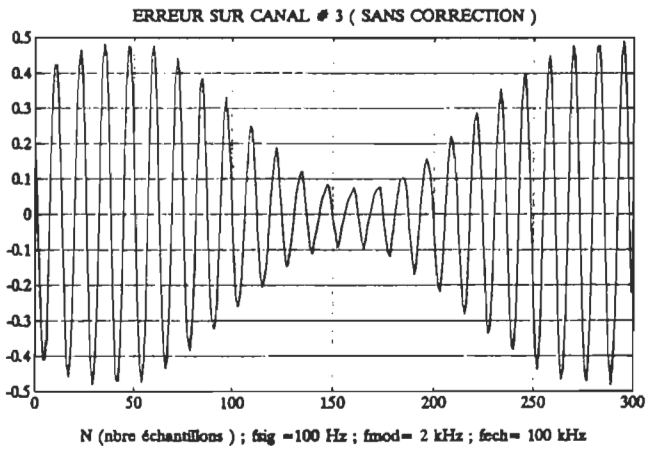


Fig.3.5-b

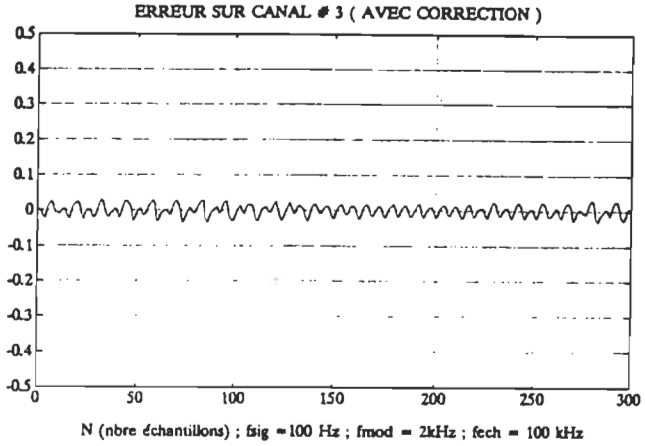


Fig.3.5-e

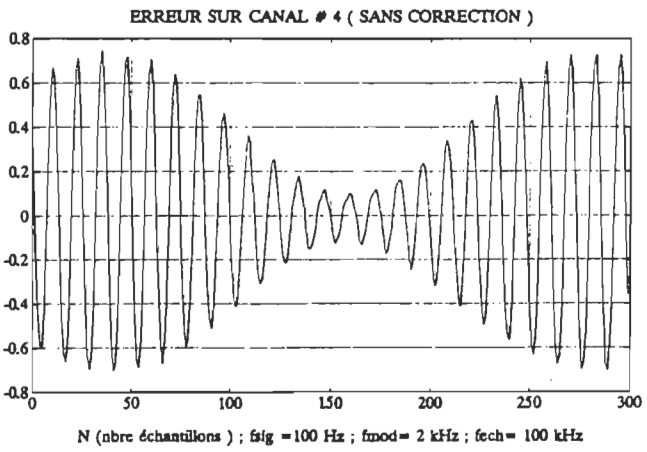


Fig.3.5-c

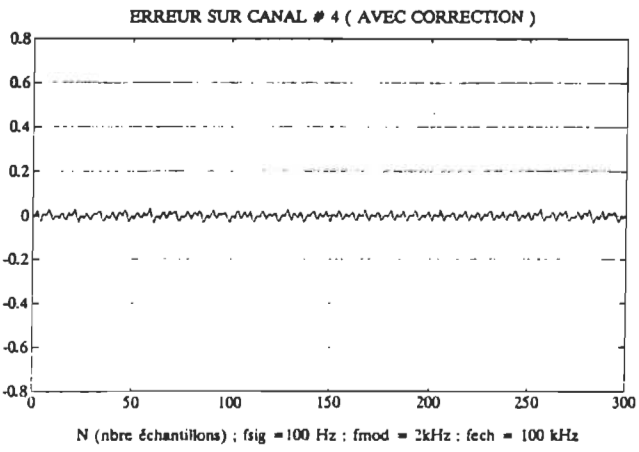


Fig.3.5-f

III-1-2-1 Correction dans le domaine fréquentiel:

Une des conclusions résultant de l'analyse fréquentielle des signaux de mesure, est l'existance d'une linéarité entre les délais des signaux ΔT_i dans le domaine temporel, et l'erreur de phase $\Delta\theta_i$, dans le domaine fréquentiel [BELLEMARE '89] :

$$\Delta\theta_i = 2\,\pi\,f\,\Delta T_i \qquad (3\text{-}2)$$

Cela signifie, que la correction dans le domaine fréquentiel sera plus facile, et sera une simple opération de soustraction de l'erreur de phase à partir du spectre de phase du signal. Il faudra donc trouver un filtre qui permet cette correction, un filtre " optimal " puisqu'il va corriger complètement la phase du signal.

L'algorithme de correction sera le suivant :

- calculer la transformée de Fourier des signaux (spectre de fréquence);
- corriger le spectre de phase en faisant la soustraction de la phase par $\Delta\theta_i$ donné par la relation (3-2) ;
- calculer la transformée de Fourier inverse et retour dans le domaine du temps.

Pour cela deux cas furent analysées :

- correction totale : $\Delta T_i = (i - 1)\,\Delta T + T_t$;
- correction partielle : $\Delta T_i = (i - 1)\,\Delta T$.

Le premier cas se réalise si on connait le temps d'acquisition total, tandis que le deuxième, plus réaliste, synchronise tous les signaux sur l'entrée #1, car généralement on ne sait pas T_t .

La méthode de correction fréquentielle s'avère trés laborieuse, elle demande beaucoup de calcul (deux transformations de Fourier- l'une directe l'autre inverse). Cette méthode n'est pas applicable si on veut que la correction se fait en temps réel.

Les filtres dans le domaine du temps peuvent donner de bons résultats et prendre moins de temps.

III-1-2-2 Correction dans le domaine temporel

III-1-2-2-a Filtre de correction optimal

La réponse impulsionnelle du filtre optimal correspond à la fonction " sinc " [SCHAFER & RABINER '73]:

$$g_i(t) = \frac{\sin(\pi B (t - \Delta T_i))}{\pi B (t - \Delta T_i)} \quad , \quad i = 1,2,3,4 \qquad (3\text{-}3)$$

où B est la largeur de bande du spectre du signal d'entrée x(t).

La méthode à utiliser maintenant pour la correction est la convolution entre cette réponse impulsionnelle et les signaux à corriger.

Mais vue la contrainte du temps de calcul, et le fait que pour avoir une correction optimale il faut utiliser plusieurs points de la fonction sinc, idéalement une infinité, cela revient encore plus vite d'utiliser la méthode fréquentielle car elle est plus rapide que la convolution.

Il faut donc un autre type de filtre capable de réaliser une bonne correction des signaux avec moins de points, donc plus rapide. Les filtres d'interpolation peuvent donner des meilleurs résultats.

III-1-2-2-b Filtres d'interpolation

Dans le domaine de l'analyse numérique, le calcul de la valeur d'une fonction entre deux points mesurés est appelé " interpolation ".

Il a été montré dans [BELLEMARE '89] que les filtres d'interpolation de Lagrange offre une bonne correction même avec un nombre de point

trés limité. Le filtre est défini par la relation :

$$y_i(n) = \sum_{\vartheta=-L}^{K} x_i(n-\vartheta) \cdot g_i(\vartheta) \qquad (3\text{-}4)$$

où :

$$g_i(\vartheta) = \prod_{\substack{j=-L \\ j\neq\vartheta}}^{K} \left(\frac{j-\Delta j}{j-\vartheta} \right) \qquad (3\text{-}5)$$

est la réponse impulsionnelle du filtre, L et K sont respectivement l'ordre inférieur et supérieur du polynôme de Lagrange utilisé pour l'interpolation, $\Delta j = \Delta T_j / T_{ac}$, avec T_{ac} le temps d'acquisition.

Dans notre cas la correction doit être faite en temps réel, c'est-à-dire en même temps qu'on fait la mesure des signaux, donc L doit être égale à 0.

Il faut trouver, maintenant, l'ordre et la caractéristique des filtres qui offrent une correction plus prés de l'optimale. Dans le cas où L = 0 , la meilleur caractéristique qui répond à nos exigences s'est trouvée pour K ≤ 2 [BELLEMARE '89]. Les coefficients du filtre correspondants dans ce cas sont déduits à partir de la relation (3-5). La réponse en fréquence des filtres est donnée aux figures 3.6 et 3.7.

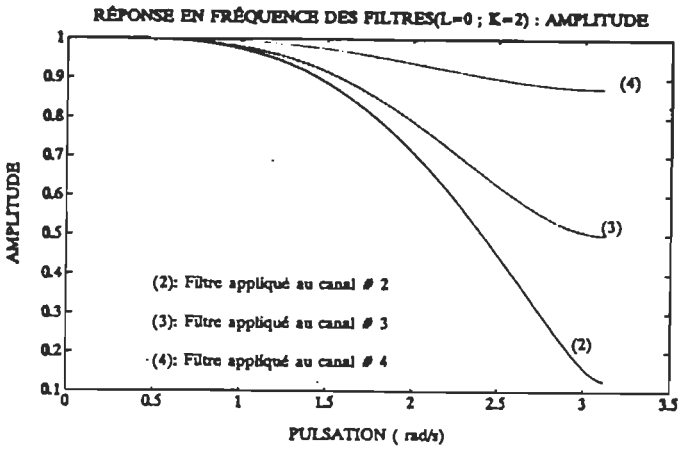


Fig.3.6

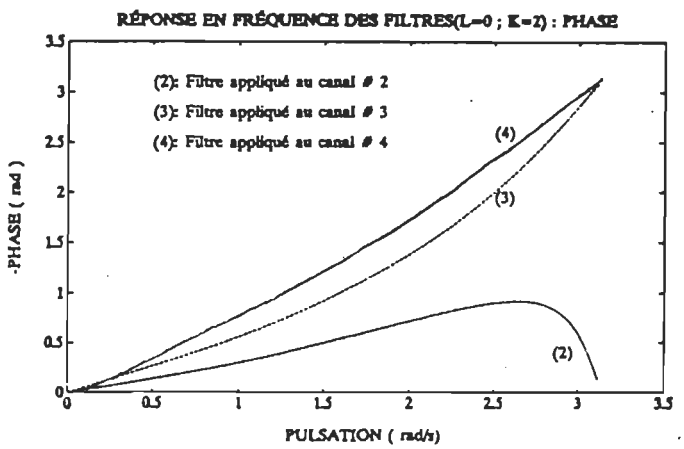


Fig.3.7

III-1-3 Implantation du filtre dans le DSP56000

La correction des erreurs consiste à la convolution des séquences discrètes des signaux d'entrées { $x_i(n)$ } avec les filtres correspondants définis par les réponses impulsionnelles { $g_i(n)$ } pour i=1,2,3,4 .

Les filtres d'interpolations de Lagrange, pour la correction des signaux, qui vont être implantés sont des filtres de type réponse impulsionnelle finie (RIF). Ils ont la forme :

$$y(n) = \sum_{k=0}^{N} g(k) \cdot x(n-k) \qquad (3\text{-}6)$$

Comme on le voit, le filtrage est effectué en accumulant les multiplications entre les échantillons de données d'entrée et les coefficients du filtre d'interpolation g(k). C'est l'équation de convolution discrète non récursive.

Grace à l'unité de multiplication-accumulation du DSP56000, la réalisation du filtre est trés facile. En effet, pour implanter ce filtre dans le DSP56000, les étapes suivantes doivent être établies:

- sauvegarder l'échantillon d'entrée, qui va être l'état initial pour la prochaine période d'échantillonnage et dans notre cas c'est x(n),
- multiplier l'entrée par g(0) et accumuler le produit des variables d'états et des coefficients,
- décaler les variables d'états pour l'échantillon prochain.

Ce processus peut être implanté sur le DSP56000 en utilisant l'addressage "modulo" pour implanter le décalage et le déplacement parallèle de données pour charger la multiplication-accumulation (MAC) du processeur.

Le schéma bloc de l'implantation matérielle pour la correction des erreurs est présenté à la figure 3.8. Le convertisseur A/N utilisé est le DSP56ADC16 de Motorola. C'est un convertisseur série de 16 bits, qui

utilise la technologie Sigma-Delta, il n'a pas besoin d'un échantillonneur-bloqueur ni d'un filtre d'antirepliement à l'entrée, car ce sont des parties inhérant de la technologie Sigma-Delta.

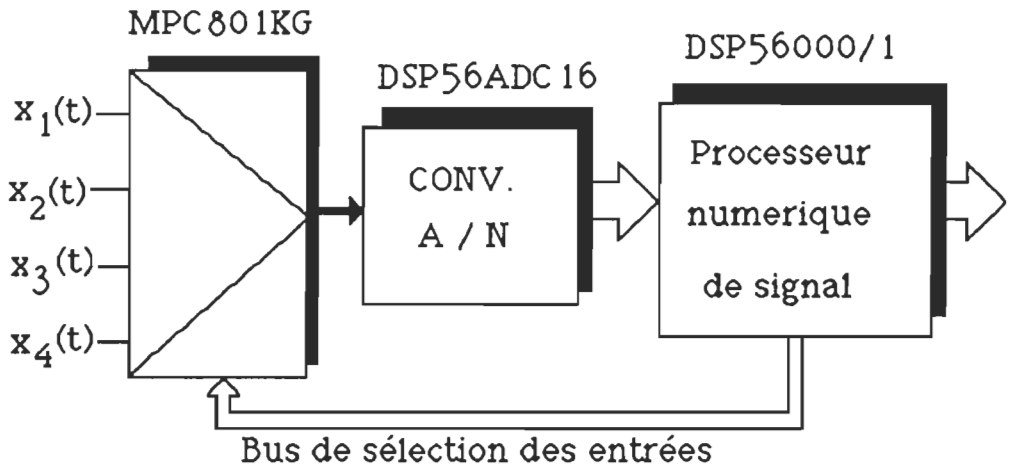


Fig 3.8 Schéma bloc d'implantation matérielle pour la correction des erreurs de multiplexage

Les figures (3.4-d),(3.4-e),(3.4-f),(3.5-d),(3.5-e) et (3.5-f) montrent les résultats pratiques de la correction des erreurs pour les deux types de signaux .

On remarque que l'erreur causée par les délais, aprés correction, est devenue trés faible par rapport à celle avant la correction. Notant ici que la correction a été faite en temps réel.

Les programmes, en langage assembleur, pour la correction des erreurs avec et sans les filtres d'interpolations de Lagrange se trouvent à l'annexe A-2.

III-2 INTERPRÉTATION DES DONNÉES SPECTROMÉTRIQUES EN UTILISANT LA MÉTHODE DE DÉCONVOLUTION SPECTRALE AVEC RÉGULARISATION DE TIKHONOV

III-2-1 Formulation du problème

On considère l'exemple illustré par la figure 3.9 qui représente le schéma simplifié d'un spectrophotomètre :

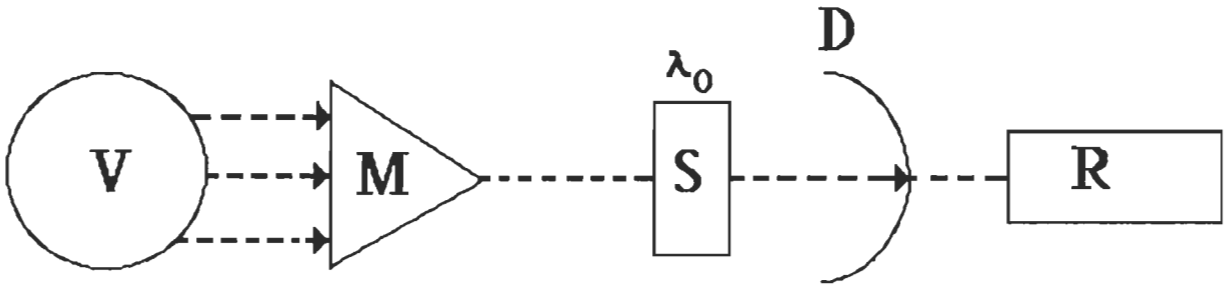


Fig.3.9 Schéma simplifié d'un spectrophotomètre

V : source de radiation visible à spectre continu pour [λmin λmax];

M : monochromateur;

S : échantillon analysé;

D : détecteur de radiation visible;

R : convertisseur de résultats de détection en signal électrique.

La source émet des radiations polychromatiques qui passent à travers M produisant un rayonnement quasi-monochromatique dont l'énergie est concentré autour de la longueur d'onde sélectionné λ_0. Le rayon arrivé à S, il sera partiellement absorbé. Son intensité, aprés S, est mesurée par le système (D+R).

À cause de la non-linéarité des appareils de mesure et de la présence des grandeurs d'influences qui perturbent les mesures, les raies du spectre des données spectrométriques se chevauchent entre eux, entrainant une fausse interprétation des résultats observés. Pour cela, il

faut essayer de reconstituer le signal de départ traduisant le signal observé, c'est la résolution du problème de déconvolution.

En effet, la relation qui relie l'entrée d'un système $x(\lambda)$ à sa sortie $y(\lambda)$, est donnée par l'intégrale de convolution

$$y(\lambda) = \int_{-\infty}^{+\infty} x(\tau)\, g(\lambda - \tau)\, d\tau = x(\lambda) * g(\lambda) \qquad (3\text{-}7)$$

où $g(\lambda)$ est la réponse impulsionnelle du système. Quand $g(\lambda)$ ne peut pas être raisonnablement approchée par une impulsion de Dirac, l'entrée $x(\lambda)$ doit être restaurée à partir de la sortie mesurée. La déconvolution, c'est à dire la solution numérique de cette équation de convolution, est un problème physique rencontré dans de nombreux domaines géophysiques, optiques, traitement de signal et des images [JANSSON'84], [HUNT '84].

Le problème auquel nous nous intéressons est celui de la déconvolution, il s'agit d'estimer les valeurs de l'entrée connaissant les valeurs passées de la sortie, en d'autre terme estimer au mieux $x(\lambda)$ à partir de $y(\lambda)$ sous divers contraintes. Pour cela, il faut analyser l'existance des solutions pour l'équation (3-7). Il faut et il suffit que $g(\lambda)$ possède un inverse de convolution noté $g^{*-1}(\lambda)$ tel que :

$$g \,.\, (g^*)^{-1} = \delta \qquad (3\text{-}8)$$

où δ est l'impulsion de Dirac. Si c'est le cas, la résolution de (3-7) est alors immédiate [BIRAUD '76].

$$x(\lambda) = g^{*-1}(\lambda) * y(\lambda) \qquad (3\text{-}9)$$

Mais le problème, c'est que $g^{*-1}(\lambda)$ n'existe pas toujours. Elle ne peut exister que pour certaines fonctions $y(\lambda)$.

Dans le domaine fréquentiel, l'équation (3-7) sera,[MAX'85] :

$$Y(j\omega) = G(j\omega) \,.\, X(j\omega) \qquad (3\text{-}10)$$

où $Y(j\omega)$, $X(j\omega)$ et $G(j\omega)$ sont les transformées de Fourier de $y(\lambda)$, $x(\lambda)$ et $g(\lambda)$ respectivement. On peut déduire donc que :

$$X(j\omega) = Y(j\omega) / G(j\omega) \qquad (3\text{-}11)$$

Pour utiliser cette égalité pour la reconstitution, il faut que $1/G(j\omega)$ existe et soit une distribution tempéré. Si $G(j\omega)$ est une fonction qui ne s'annule pour aucune valeur de fréquence f et qui ne tende pas vers zéro à l'infini plus vite qu'une puissance de 1/f alors on peut définir 1/G(f) de façon unique. Dans la pratique ces conditions ne sont jamais réalisées. En optique, en radioastronomie, par exemple, $G(j\omega)$ est une fonction qui, théoriquement, s'annule au dela d'une fréquence de coupure f_{co}. Mais dans la majorité des cas pratiques $|G(j\omega)| \dashrightarrow 0$ quand f croit. Même dans ce cas, $G(j\omega)$ étant plus souvent affecté d'un bruit de mesure η, il existera une certaine fréquence f_0 au dela de laquelle on aura :

$$|G(j\omega)| \approx \sigma_\eta \qquad (3\text{-}12)$$

où σ_η^2 est la variance du bruit η.

Alors $G(j\omega)$ présentera de multiple passage par zéro et la division $1/G(j\omega)$ est pratiquement impossible. En effet, l'unicité des solutions n'est jamais démontrable dans le cas pratique.

Pour illustrer ce qu'on vient de dire, on va supposer qu'on a enregistré simultanément le signal d'entrée, le signal de sortie et qu'on connait d'avance sa réponse impulsionnelle. Au signal de sortie on ajoute un bruit aléatoire trés faible (car en général on ne mesure jamais un signal excate mais toujours un signal entaché d'une erreur ou bruit) et à l'aide des relations (3-10) et (3-11) on essaie de retrouver le signal $x(\lambda)$ à partir de $y(\lambda)$ et $g(\lambda)$, en utilisant la formule suivante :

$$\hat{X}(j\omega) = \frac{\tilde{Y}(j\omega)}{G(j\omega)} \quad \Longrightarrow \quad x(\lambda) = \mathcal{F}^{-1}(\hat{X}(j\omega)) \qquad (3\text{-}13)$$

où $\tilde{Y}(j\omega)$ - la transformée de Fourier du signal (y(t) + bruit).

Les signaux $x(\lambda)$ et $g(\lambda)$ (fig.3.10 et fig.3.11) sont des signaux de type spectrométriques, donnés par les expressions suivantes :

$$x(\lambda) = \left(\frac{e^{-(\lambda-.8)^2/.03} + e^{-(\lambda-1.2)^2/.03}}{.9550408} - .052130913 \right) 1.4\,\lambda$$

pour $.5 < \lambda < 1.5$

$= 0$ ailleurs (3-14)

$$g(\lambda) = 6\, e^{-80\,\lambda^2}$$ pour $-.5 < \lambda < .5$

$= 0$ ailleurs (3-15)

Le bruit ajouté au résultat de convolution (fig.3.12) est de distribution normale et de variance $\sigma_\eta^2 = 10^{-10}$.

Le résultat de reconstitution obtenu, par simulation, est présenté à la figure 3.13. Lors de la simulation on a supposé que les données sont discrètes, et l'opération de convolution a été calculée par le produit des transformées de Fourier discrètes des séquences $\{x_n\}$ et $\{g_n\}$.

On remarque que la solution ne converge pas vers le signal exacte, on voit des oscillations qui ne s'amortissent pas, et l'erreur efficace de reconstitution

$$\varepsilon[x] = \sqrt{\frac{1}{N}\sum_{n=0}^{N-1} [\hat{x}(n) - x(n)]^2} = 4.48\ 10^{+5} \qquad (3\text{-}16)$$

est énorme (N est le nombre de points de discrétisation du signal $x(\lambda)$).

Il s'agit donc d'un problème mal conditionné vu que le résultat de reconstitution est trés sensible aux bruits qui affectent le signal. En conséquence, il doit être régularisé afin de diminuer l'influence du bruit dans la solution finale.

En l'abscence de ce bruit, la formule utilisée devra donner la reconstitution excate du signal réel. Mais dans la pratique, les résultats sont bruités et l'on ne peut envisager de solutions qu'a l'aide de contraintes physiques et externes, qui sont déterminées par les

informations a priori que l'utilisateur possède sur le système de mesure. L'existance de ces contraintes assure l'obtention d'une solution minimisant un critère d'erreur donné, permettant de réduire l'influence du bruit dans la solution finale [BIRAUD '76],[SABATIER '83] .

La méthode de déconvolution spectrale avec la régularisation de Tikhonov apparait parmis les méthodes qui offre une solution trés efficace pour la résolution du problème de déconvolution [MIEKINA & MORAWSKI '86].

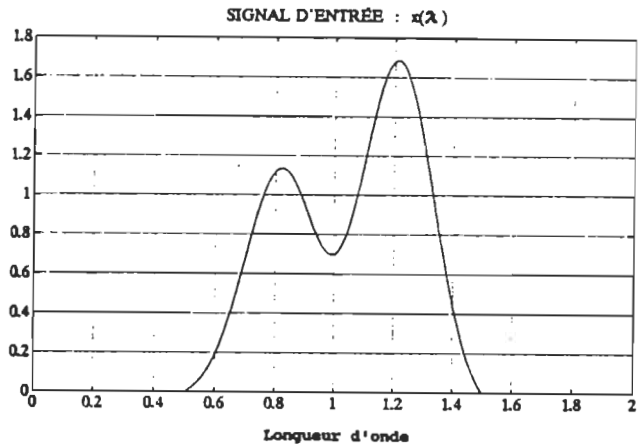


Fig.3.10

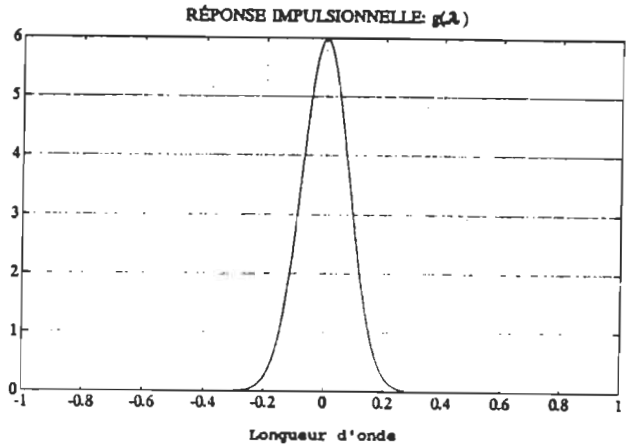


Fig.3.11

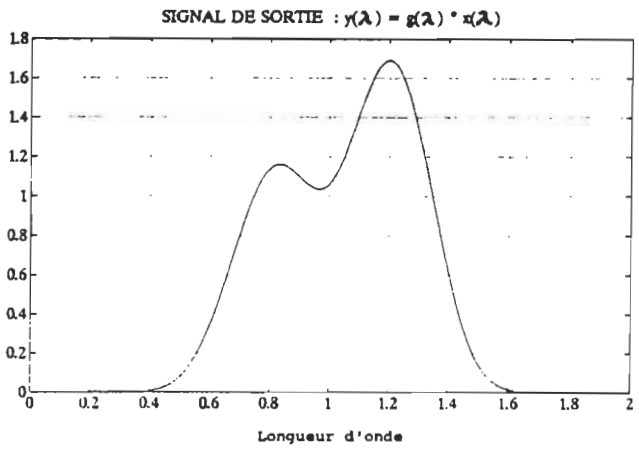


Fig.3.12

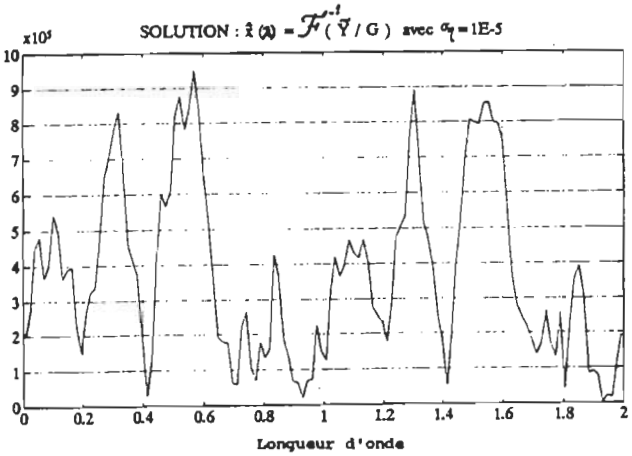


Fig.3.13

III-2-2 Méthode de déconvolution spectrale avec régularisation de Tikhonov

Pour reconstituer un signal, on a recourt à un modèle mathématique qui relie le signal d'entrée x au signal de sortie y.

$$\mathcal{G}\,[\,x\,] = y \qquad (3\text{-}17)$$

où $\mathcal{G}$ est un opérateur supposé connu a priori, les signaux x et y appartiennent à l'espace Hilbert Hx et Hy, repectivement.

Cette équation doit être résolue suivant le signal x et à partir des données de mesure y. Or ces dernières sont discrètes, elles sont donc sujet à des erreurs de discrétisation et autres. On doit donc définir les limites supérieures de ces erreurs qui représentent les distorsions de ses données à savoir :

$$\| \tilde{y} - y \|_{Hy} \leq \Delta_Y \qquad (3\text{-}18)$$

La méthode de régularisation de Tikhonov décrite dans [MIEKINA & MORAWSKI '86] est basée sur la minimisation du critère [TIKHONOV'76]:

$$J^{\alpha}[\,x\,] = \| \mathcal{G}(x) - \tilde{y} \|^2_{Hy} + \alpha \| x \|^2_{Hx} \qquad (3\text{-}19)$$

où $\alpha > 0$ dit paramètre de régularisation et dont sa valeur optimale doit être déterminée selon le niveau d'erreur Δ_Y. Il est déterminé selon la procédure suivante:

1º définir la fonction $\rho(\alpha)$:

$$\rho(\alpha) = \| \mathcal{G}(x^{\alpha}) - \tilde{y} \|^2_{Hx \to Hy} - \Delta^2_Y \qquad (3\text{-}20)$$

où x^{α} représente la solution correspondante au paramètre de régularisation α.

2º **si** $\| \tilde{y} \|_{Hy} > \Delta_Y$ =====>

- si $\rho(\alpha) > 0$ pour tout $\alpha > 0$ ===> α ----> 0^+

- autrement α est la racine positive de l'équation $\rho(\alpha) = 0$.

3º **si** $\| \tilde{y} \|_{Hy} < \Delta_Y$ =====>

la solution optimale est $x = 0$ et α ne sera pas calculé.

III-2-3 Développement de l'algorithme de déconvolution spectrale avec la régularisation de Tikhonov

Pour faciliter le problème on va considérer le cas où le modèle mathématique est une équation intégrale de convolution, $\mathcal{G}[x] = g * x$ définie dans les espaces $Hx=W_2^1$ (espace Sobolev) et Hy=L2. Le modèle de données sera le suivant :

$$\tilde{y}(\lambda) = (g * x)(\lambda) + \eta(\lambda) \qquad \text{pour } \lambda \in [0 \ \Lambda] \quad (\Lambda=\lambda_{max} - \lambda_{min})$$
$$= 0 \qquad \text{ailleurs} \qquad (3\text{-}21)$$

où $\eta(\lambda)$ est une réalisation d'un processus aléatoire ergodique $\mathfrak{n}(\lambda)$ avec $E[\mathfrak{n}(\lambda)] = 0$ et $Var[\mathfrak{n}(\lambda)] = \sigma_\eta^2$. En conséquence, $\tilde{y}(\lambda)$ est une réalisation d'un processus aléatoire

$$\tilde{\mathfrak{y}}(\lambda) = (g * x)(\lambda) + \mathfrak{n}(\lambda) \qquad \text{pour } \lambda \in [0 \ \Lambda] \qquad (3\text{-}22)$$

Selon l'approche des moindres carrées, une approximation optimale du signal reconstitué , noté $\hat{x}(\lambda)$, doit minimiser la norme $\| \tilde{y} - g*x \|_2^2$. Une telle solution du problème, dans plusieurs cas, n'est pas suffisamment stable. En conséquence, l'idée de l'approche alternative basée sur la minimisation de la norme de x dans l'espace Sobolev

$$J[\hat{x}] = \| \hat{x} \|_{2,1}^2 \text{ ----------> MINIMALE} \qquad (3\text{-}23)$$

et sur l'exigence qu'une estimé a priori de la variance σ_η^2 , notée $\hat{\sigma}_\eta^2$, est accessible tel que

$$\| \tilde{y} - g * x \|_2^2 = \Lambda \hat{\sigma}_\eta^2 \qquad (3\text{-}24)$$

semble être relativement bonne puisqu'elle utilise plus d'information a

priori $\hat{\sigma}_\eta{}^2$ qui modélisent les erreurs de mesure. De plus elle permet d'éliminer les caractéristiques indésirables des solutions obtenues sans régularisation. La relation (3-24) est justifiée par l'égalité suivante, qui est valable pour un processus ergodique $\eta(\lambda)$

$$\sigma_\eta^2 = \mathrm{Var}[\,\eta(\lambda)\,] \approx \frac{1}{\Lambda}\int_0^\Lambda [\,\eta(\lambda)\,]^2\, d\lambda \approx \frac{1}{\Lambda}\int_0^\Lambda [\tilde{y}(\lambda) - (g*x)(\lambda)]^2\, d\lambda$$

$$\approx \frac{1}{\Lambda} \| \tilde{y} - g * x \|_2^2 \qquad (3\text{-}25)$$

et à partir de cette relation et la relation (3-20), on peut déduire que

$$\Delta_Y = \sqrt{\Lambda}\ \hat{\sigma}_\eta \qquad (3\text{-}26)$$

Alors la solution proposée a la forme

$$x(\lambda) = \arg_x \inf \{ \| x \|_{2,1}^2 \ \Big|\ \| \tilde{y} - g * x \|_2^2 = \Lambda \hat{\sigma}_\eta^2 \} \qquad (3\text{-}27)$$

Le problème peut être plus facilement à résoudre dans le domaine fréquentiel. En utilisant l'identité de Parseval [MAX '85], on obtient

$$\| \hat{x} \|_{2,1}^2 = \frac{1}{2\pi}[\, \| \hat{X} \|_2^2 + \omega^2 \| \hat{X} \|_2^2 \,] = \frac{1}{2\pi} \| (1 + \omega^2)\, \hat{X} \|_2^2 \qquad (3\text{-}28)$$

$$\| \tilde{y} - g * \hat{x} \|_2^2 = \frac{1}{2\pi} \| \tilde{Y} - G \hat{X} \|_2^2 \qquad (3\text{-}29)$$

où $\hat{X} = \hat{X}(j\omega) = \mathcal{F}\{ x(\lambda) \}$

$\tilde{Y} = Y(j\omega) = \mathcal{F}\{ y(\lambda) \}$

$G = G(j\omega) = \mathcal{F}\{ g(\lambda) \}$

Alors l'équation (3-25) sera

$$\| \tilde{Y} - G \hat{X} \|_2^2 = 2\pi\, \Lambda \hat{\sigma}_\eta^2 \qquad (3\text{-}30)$$

et la solution dans le domaine fréquentiel est

$$X = \arg_X \inf \{ \alpha \| (1 + \omega^2)\, \hat{X} \|_2^2 + \| \tilde{Y} - G X \|_2^2 \ \Big|\ \| \tilde{Y} - G\hat{X} \|_2^2 = 2\pi\Lambda\hat{\sigma}_\eta^2 \} \qquad (3\text{-}31)$$

où $\alpha > 0$ est le paramètre de régularisation. Cette solution doit satisfaire

l'équation d'Euler-Lagrange, à savoir, [HAMMING'73 § 43] :

$$\frac{\partial F(\hat{X})}{\partial X} = 0 \Leftrightarrow \frac{\partial F(\hat{X})}{\partial X_R} + j \frac{\partial F(\hat{X})}{\partial X_I} = 0 \qquad (3\text{-}32)$$

$$\Leftrightarrow \frac{\partial F(\hat{X})}{\partial X_R} = 0 \quad \text{et} \quad \frac{\partial F(\hat{X})}{\partial X_I} = 0$$

où

$$F(\hat{X}) = \alpha(1 + \omega^2) \mid \hat{X} \mid^2 + \mid \tilde{Y} - G\hat{X} \mid^2 \qquad (3\text{-}33)$$

X_R et X_I : partie réelle et imaginaire de X.

Aprés quelque reformulation, on aboutit à l'expression suivante :

$$[\alpha(1 + \omega^2) + \mid G \mid^2] \hat{X} - \tilde{Y} G^* = 0 \qquad (3\text{-}34)$$

$$\Rightarrow \qquad \hat{X} = \frac{\tilde{Y} G^*}{\mid G \mid^2 + \alpha(1 + \omega^2)} \qquad (3\text{-}35)$$

Cette dernière relation est trés importante car elle nous permet par une transformation de Fourier inverse de retrouver le signal d'entrée à partir des données du signal de sortie ainsi que de la réponse impulsionnelle g.

Aprés substitution de la relation (3-35) dans la relation (3-30), on peut écrire:

$$\left\| \frac{\alpha(1 + \omega^2)\, \tilde{Y}}{\mid G \mid^2 + \alpha(1 + \omega^2)} \right\|_2^2 = 2\pi \Lambda \hat{\sigma}_\eta^2 \qquad (3\text{-}36)$$

qui peut être résolue selon la procédure décrite dans § III-2-2, utilisant une méthode numérique tel que la méthode de Newton d'approximation successive [BLUM '72] pour trouver la valeur optimale α qui offre une meilleur estimation du signal reconstitué.

Le premier problème à résoudre consiste à la discrétisation des relations (3-35) et (3-36). Pour cela on suppose que l'axe λ a été normalisé dans le sens que $\lambda := \lambda - \lambda_{min}$; en conséquence $\lambda_{min}=0$ et $\lambda_{max}=\Lambda$. On suppose aussi que les spectres $|\hat{X}(j\omega)|$, $|G(j\omega)|$ et $|\tilde{Y}(j\omega)|$ sont

negligeables pour $| \omega | \geq \Omega_{max}$ où Ω_{max} est connu. Si le pas d'échantillonnage $\Delta\lambda$, le long de l'axe λ, satisfasse la condition de Shannon, à savoir

$$\Delta\lambda \leq \pi / \Omega_{max} \tag{3-37}$$

les spectres continus $\hat{X}(j\omega)$, $G(j\omega)$ et $\tilde{Y}(j\omega)$ peuvent être approximés en utilisant les représentations discrètes de ces signaux. Pour $x(\lambda)$ on a

$$\hat{X}(j\omega_k) = \int_0^{\Lambda} \hat{x}(\lambda)\, e^{-j\omega_k \lambda}\, d\lambda \approx \Delta\lambda \sum_{n=0}^{N-1} x_n\, e^{-jkn\Delta\lambda\Delta\omega} \tag{3-38}$$

où $\Delta\lambda = \Lambda/N$, $\omega_k = k\,\Delta\omega$ pour k=0,1,...,N-1 et $\Delta\omega$ le pas d'échantillonnage le long de l'axe ω. Dans les règles d'utiliser les algorithmes de FFT discrètes (DFT) on doit choisir

$$\Delta\omega\, \Delta\lambda = 2\pi/N \tag{3-39}$$

donc

$$\hat{X}(j\omega_k) = \Delta\lambda \sum_{n=0}^{N-1} x_n\, e^{-jkn\frac{2\pi}{N}} = \Delta\lambda\, \hat{X}_k \tag{3-40}$$

où $\{ \hat{X}_k \} = DFT\{ x_n \}$. Pour cette valeur de $\Delta\omega$, on obtient conformément à la relation (3-37)

$$N\, \Delta\omega = 2\pi/\Delta\lambda \geq 2\pi\, \Omega_{max} /\pi = 2\Omega_{max} \tag{3-41}$$

indiquant que la plage des fréquences discrètes $\{\omega_k\}$ couvre les largeurs de bandes des signaux $\hat{x}(\lambda)$, $g(\lambda)$ et $\tilde{y}(\lambda)$. De la même façon, on obtient

$$G(j\omega_k) = \Delta\lambda\, G_k \qquad \tilde{Y}(j\omega_k) = \Delta\lambda\, \tilde{Y}_k \tag{3-42}$$

Par subtitution des ses relations dans les relations (3-35) et (3-36), on aura

$$\hat{X}_k = \frac{\Delta\lambda\, (G_k)^*\, \tilde{Y}_k}{|\Delta\lambda\, G_k|^2 + \alpha\left(1 + k^2 \left(\frac{2\pi}{\Lambda} \right)^2 \right)} \tag{3-43}$$

et

$$\sum_{n=0}^{N-1} \left\{ \frac{\alpha\left(1 + k^2\left(\frac{2\pi}{\Lambda}\right)^2\right) |\Delta\lambda\, \tilde{Y}_k|^2}{|\Delta\lambda\, G_k|^2 + \alpha\left(1 + k^2\left(\frac{2\pi}{\Lambda}\right)^2\right)} \right\}^2 = \Lambda\, \hat{\sigma}_\eta^2 \qquad (3\text{-}44)$$

III-2-4 Résultats de simulations

L'étude de l'algorithme est essentiellement portée à l'evaluation de son exactitude, en particulier, la sensibilité de l'erreur efficace par rapport au nombre de point de discrétisation N et au niveau d'erreur Δ_Y estimé a priori.

Pour la simulation, les essais ont été faites selon la méthodologie suivante:

1º Une solution exacte $x(\lambda)$ a été supposée, qui a la même forme que la caractéristique réelle S-M-D (fig.3.10), donnée par la relation (3-14).

2º Le signal mesuré $y(\lambda)$ (fig.3.12) est déterminé à partir de la convolution de $x(\lambda)$ et une fonction $g(\lambda)$ (fig.3.11) qui traduit le modèle approximatif d'entrée/sortie du propriété du monochromateur, donnée par la relation (3-15). Les données ont été supposées discrètes pour toutes les opérations de l'algorithme.

3º À $y(\lambda)$ on ajoute un signal aléatoire "bruit blanc", dont la valeur moyenne est nulle et de variance σ_η^2.

Pour la détermination du paramètre de régularisation α, on a utilisé la méthode de Newton d'approximation successive, et pour la reconstitution du signal d'entrée la relation (3-35) a été utilisée, l'erreur de reconstitution est déterminée à partir de la relation (3-16).

Les résultats de simulations, obtenus pour différentes valeurs de variance du bruit σ_η^2 et pour différentes estimés a priori du niveau d'erreur Δ_Y, sont présentés aux tableaux 3.2-I, 3.2-II et 3.2-III

(résultats de simulations).

Les résultats de reconstitution graphiques sont présentés aux figures 3.14-a , 3.14-b, 3.15-a, 3.15-b, 3.16-a, 3.16-b et 3.17.

On peut dire, en regardant aux résultats de simulation, que la méthode de régularisation de Tikhonov est un outil effectif pour la réduction de l'effet du bruit quand elle est appliquée au problème de déconvolution. Cependant, son exactitude dépend de la crédibilité de l'information a priori du bruit affectant les données de mesure, Δ_Y. En effet, l'exactitude de reconstitution du signal est sensible à la sous-estimation de Δ_Y que sur son sur-estimation (fig.3.18). En d'autre terme il existe une valeur Δ_Y optimum pour chaque variance du bruit correspondante à une erreur de reconstitution $\varepsilon[x]$ optimum (fig.3.19 et fig.3.20).

1°- TABLEAU 3.2-I : Résultats de simulations avec $\sigma_\eta = .001$ et N=128

Δ_Y	ϵ [x]	α
1.10E-03	60.52	2.4100E-12
1.20E-03	1.2680E-02	1.2810E-04
1.22E-03	4.6537E-03	2.1695E-03
1.28E-03	3.0089E-03	1.2864E-02
1.30E-03	2.9980E-03	1.5587E-02
1.32E-03	3.0050E-03	1.8020E-02
1.38E-03	3.0510E-03	2.4230E-02
1.50E-03	3.1570E-03	3.4177E-02
3.00E-03	4.5780E-03	1.5100E-01
1.00E-02	1.3000E-02	4.5140E-01
5.00E-02	5.8780E-02	3.2690E+00

2°- TABLEAU 3.2-II : Résultats de simulations avec σ_η =.01 et N=128

Δ_Y	ϵ [x]	α
1.10E-02	605	2.4100E-12
1.25E-02	1.1645E-02	4.7539E-02
1.30E-02	8.1696E-03	1.4350E-01
1.34E-02	7.7700E-03	1.9600E-01
1.35E-02	7.7630E-03	2.0787E-01
1.36E-02	7.7790E-03	2.1900E-01
1.40E-02	8.0180E-03	2.6100E-01
1.50E-02	9.2280E-03	3.5230E-01
1.70E-02	1.2300E-02	5.0800E-01
3.00E-02	3.0600E-02	1.1400E+00

3°- TABLEAU 3.2-III : Résultats de simulations avec σ_η =.1 et N=128

Δ_Y	ϵ [x]	α
1.30E-01	4.6500E-02	1.1245
1.31E-01	4.5150E-02	1.3180
1.32E-01	4.4595E-02	1.5093
1.33E-01	4.4630E-02	1.6975
1.35E-01	4.5940E-02	2.0690
1.50E-01	6.9790E-02	4.9440
2.00E-01	1.3800E-01	19.4890

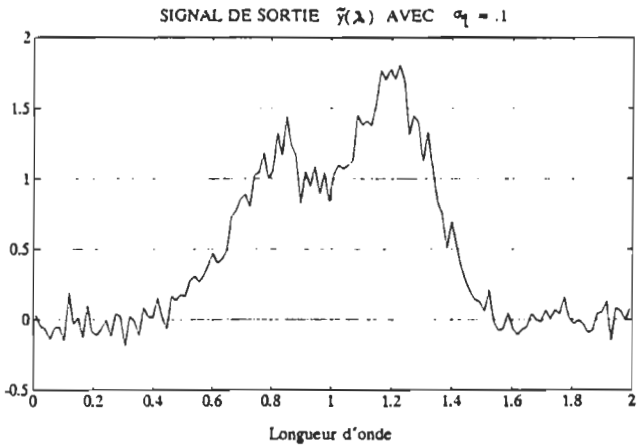


Fig.3.14-a

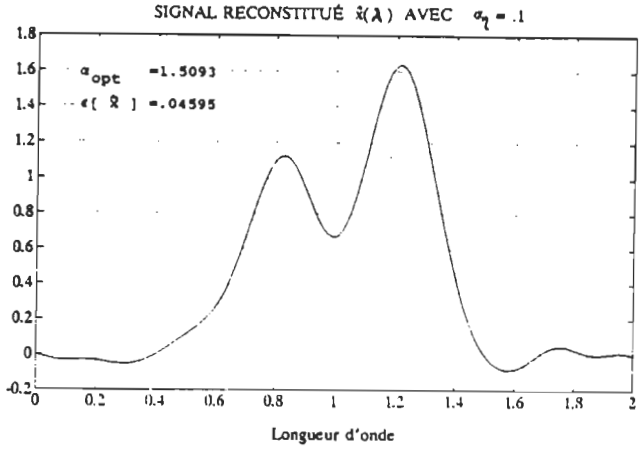


Fig.3.14-b

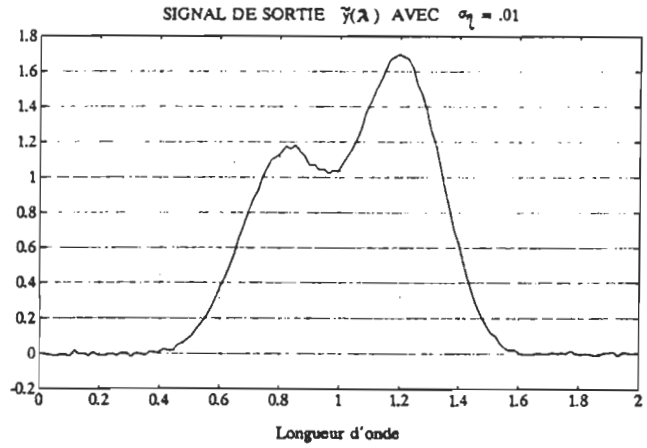


Fig.3.15-a

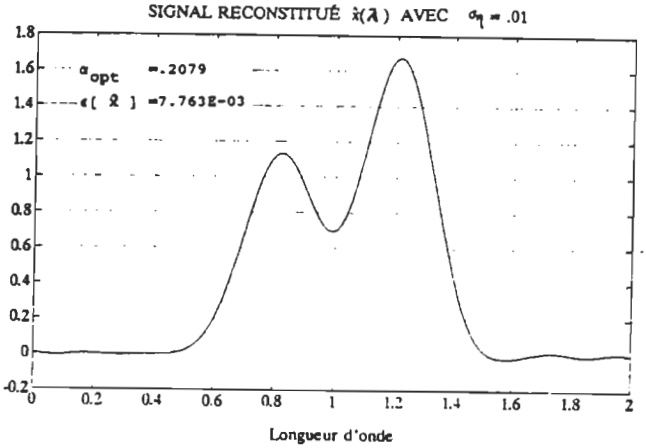


Fig.3.15-b

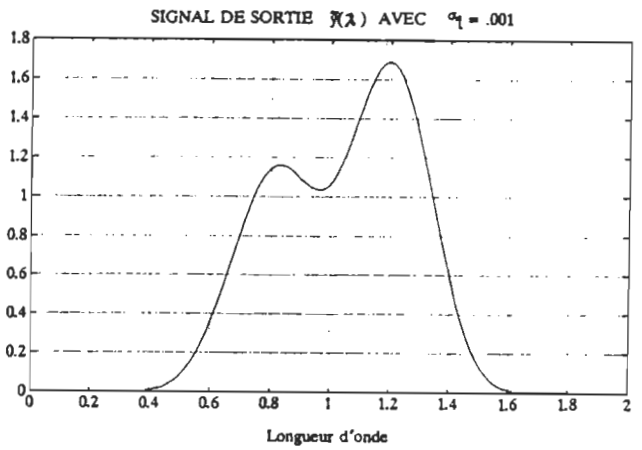


Fig.3.16-a

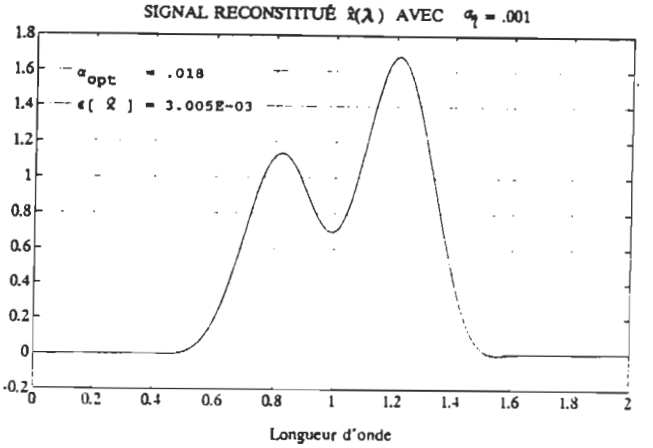


Fig.3.16-b

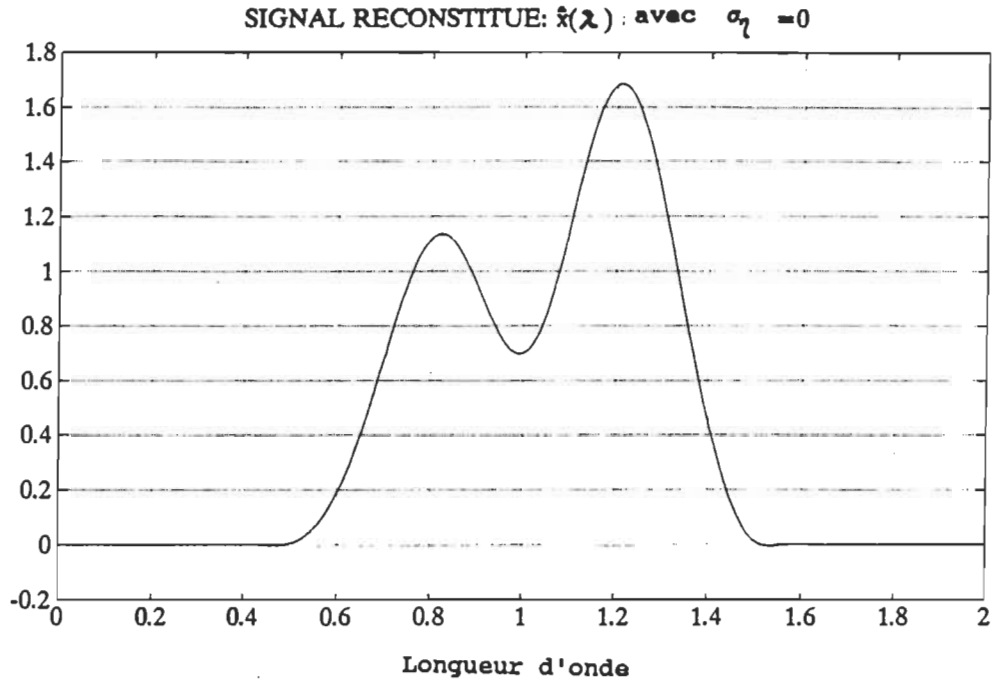


Fig.3.17

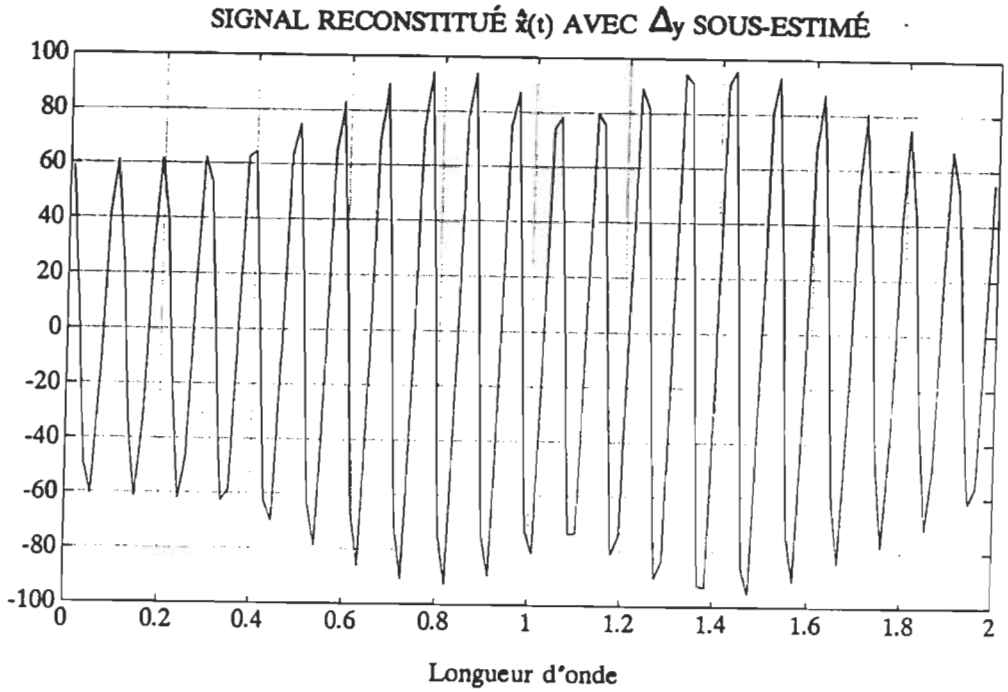


Fig.3.18

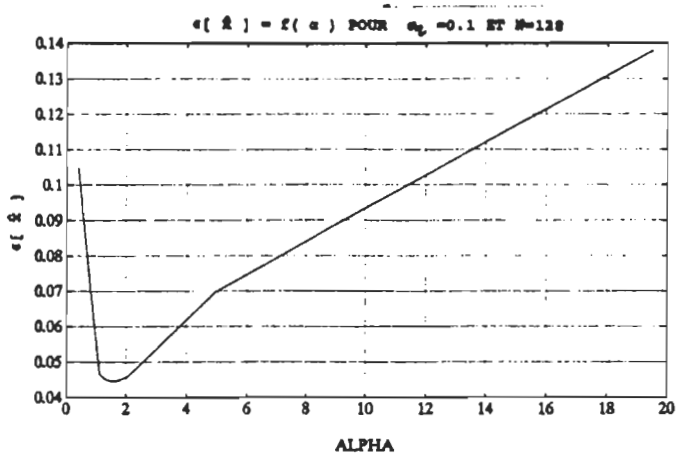


Fig.3.19-a

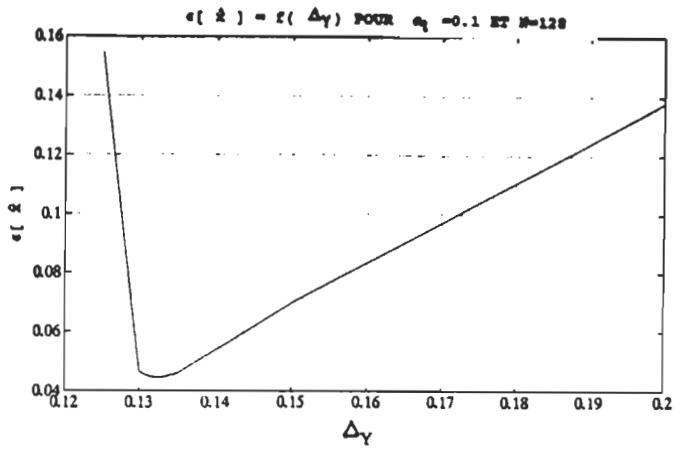


Fig.3.20-a

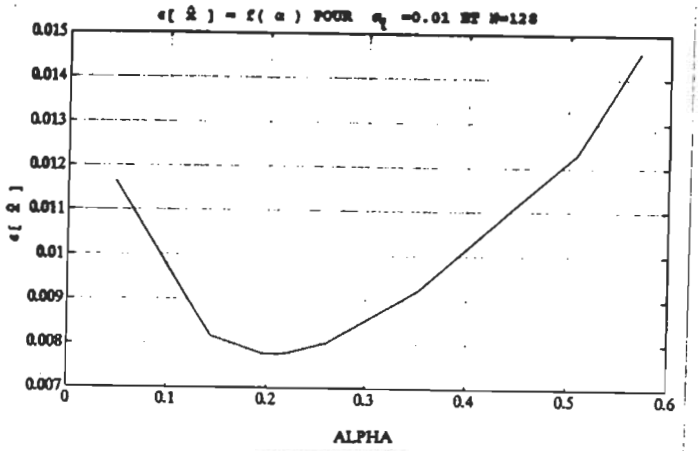


Fig.3.19-b

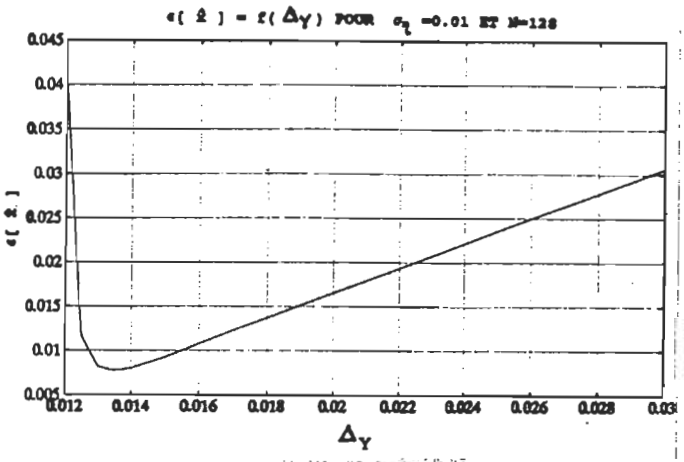


Fig.3.20-b

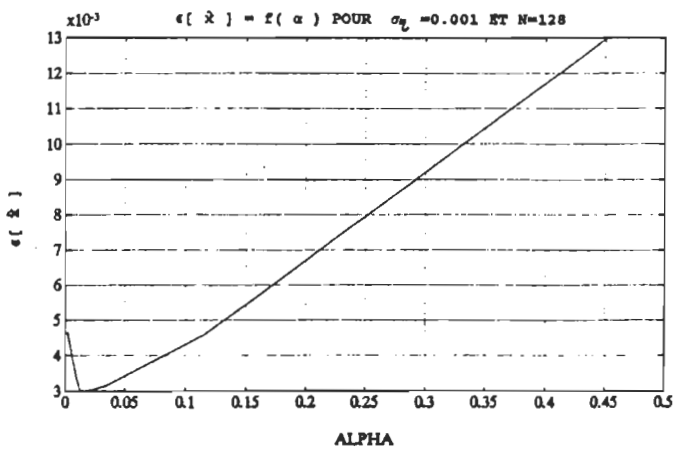


Fig.3.19-c

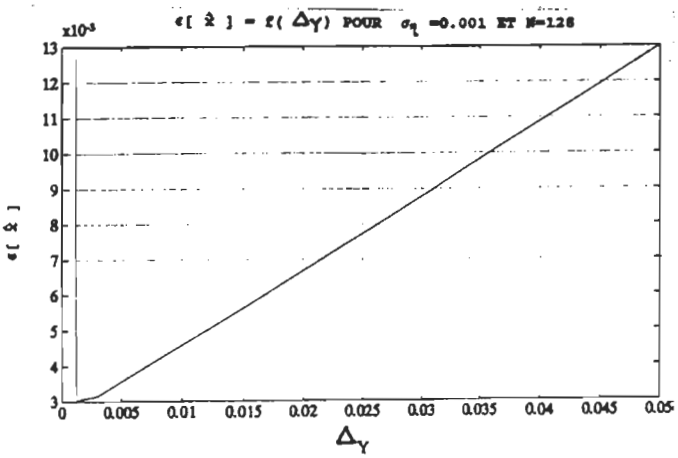


Fig.3.20-c

III-2-5 Etude de l'implantation matérielle de l'algorithme de déconvolution spectrale sur le DSP56000

La méthode de déconvolution développée dans le paragraphe précédent présente une certaine particularité qui est la grande quantité de calcul arithmétique nécessaire pour reconstituer le signal $x(\lambda)$ (relation (3-35) et (3-36)). Il s'en suit l'importance d'utilisation d'un processeur qui possède une grande capacité de calcul avec une meilleur exactitude de traitement de données. En effet, l'exactitude des données de mesure est trés importante pour avoir une erreur de reconstitution minimale. Pour cela on procède à l'étude des différentes sources d'erreurs qui peuvent influencer la reconstitution du mesurande.

III-2-5-1 Sources de transmission des erreurs

Celon l'algorithme, les données qui sont sujet à des erreurs sont ,surtout, les signaux $y(\lambda)$ et $g(\lambda)$. Ces données sont, généralement, discrètes et en nombre fini. Ils sont, alors, l'origine de l'existance des sources d'erreurs qu'il faut tenir compte lors de l'implantation de l'algorithme de déconvolution. En effet, on distingue deux types de sources d'erreurs :

- erreurs causées suite à l'acquisition des données par le spectromètre,
- erreurs dues à la représentation des données par un nombre de bits fini.

Pour l'étude de la première source d'erreurs, on va considérer que les données de mesure varient chacune de 1% et on essaie de voir l'influence de cette variation sur l'erreur de reconstitution $\varepsilon[\, x \,]$.

* Pour le signal $y(\lambda)$

Soit $\overset{\circ}{x}$ le signal reconstitué supposé exacte ($\pm\, \varepsilon\, [\, \overset{\circ}{x} \,]$), avec

$$\mathring{x} = f (\mathring{y}_1, \mathring{y}_2, \ldots, \mathring{y}_N) \qquad (3\text{-}45)$$

$\mathring{y}_1, \mathring{y}_2, \ldots, \mathring{y}_N$ la séquence des résultats de convolution supposé aussi exacte. On change , aprés, une valeur de la séquence y de 1 % , par la suite on aura une autre relation :

$$\hat{x}_n = f (\mathring{y}_1, \mathring{y}_2, \ldots, \tilde{y}_n, \ldots, \mathring{y}_N) \qquad (3\text{-}46)$$

avec $\tilde{y}_n = \mathring{y}_n + 0.01.\mathring{y}_n = 1.01\ \mathring{y}_n \qquad (3\text{-}47)$

on aura de nouveau un autre signal reconstitué et une nouvelle valeur de l'erreur $\varepsilon[\ \hat{x}\ ]$. Le coefficient de transmision de l'erreur dû à cette variation est défini par l'expression suivante :

$$T_n = \frac{\dfrac{\varepsilon [\ \hat{x}\ ] - \varepsilon [\ \mathring{x}\ ]}{\varepsilon [\ \mathring{x}\ ]}}{\dfrac{\tilde{y}_n - \mathring{y}_n}{\mathring{y}_n}} \qquad n = 1, 2, \ldots, N \qquad (3\text{-}48)$$

où $\varepsilon\ [\ \hat{x}\ ]$, $\varepsilon\ [\ \mathring{x}\ ]$ représentent l'erreur de reconstitution avec et sans erreurs de données respectivement. Pour la détermination de T_n, on a utilisé les hypothèses suivantes :

- la valeur du paramètre de régularisation, α, est constante ,
- les données utilisées sont sans présence du bruit.

Les résultats obtenus, sont représentés par les courbes $T_n = f(n)$ pour différentes valeurs de N . (fig.3.21-a, fig.3.21-b et fig.3.21-c)

* Pour le signal $g(\lambda)$

Dans ce cas on a aussi

$$\mathring{x} = f (\mathring{g}_1 , \mathring{g}_2, \ldots, \mathring{g}_N) \qquad (3\text{-}49)$$

où $\mathring{g}_1 , \mathring{g}_2, \ldots, \mathring{g}_N$ représente la séquence de données de la réponse impulsionnelle $g(\lambda)$ supposé exacte.

et $\hat{x}_n = f (\mathring{g}_1, \mathring{g}_2, \ldots, \tilde{g}_n, \ldots, \mathring{g}_N) \qquad (3\text{-}50)$

avec $\tilde{g}_n = \mathring{g}_n + 0.01.\mathring{g}_n = 1.01\ \mathring{g}_n \qquad (3\text{-}51)$

L'expression de T_n est la même que précédemment. Les résultats graphiques obtenus sont présentés aux fig.3.22-a, 3.22-b et 3.22-c. On

remarque que :

- les coefficients de transmission des erreurs T_n sont proportionnels aux séquences de données $\{y_n\}$ et $\{g_n\}$ n=1,2,...,N et ils sont trés faibles pour les variations de données sur g_n que pour celles de y_n .

- quand le nombre de points de discrétisation N augmente les coefficients T_n diminuent .

Cette analyse nous confirme que l'erreur globale de reconstitution est influencée par les erreurs de données sur $y(\lambda)$ beaucoup plus que par celles de $g(\lambda)$.

L'étude de la deuxième sources d'erreurs consiste à analyser l'effet de la représentation numérique des données, par un nombre de bits fini, sur l'erreur totale de reconstitution. Ces erreurs interviennent, surtout, dans les algorithmes de FFT et les autres opérations classiques de l'algorithme. Pour cela, on doit déterminer le nombre de bits nécessaire pour notre traitement de données qui guarantie une exactitude acceptable.

On défini le paramètre suivant:

$$eps = 5 . 10^{-m} \qquad (3\text{-}52)$$

où m désigne le nombre de chiffre de mantisse nécessaire pour représenter les données. Pour la détermination des coefficients de transmission des erreurs dus aux algorithmes de FFT sur les données $y(\lambda)$ et $g(\lambda)$, on a utilisé l'algorithme simplifié de " Cooley-Tukey "[LIM & OPPEINHEIM '88] . On pose alors :

Cy le coefficient de transmission des erreurs pour $y(\lambda)$,

Cg le coefficient de transmission des erreurs pour $g(\lambda)$.

À partir de la relation (3-35), on détermine le coefficient de transmission d'erreur C_x pour le signal $x(\lambda)$, soit :

$$|C_x|_{Max} = \frac{1}{K^2}((|\tilde{Y}|C_g + |G^*|C_y)K + |\tilde{Y}||G^*|(2|G|C_g + 2\alpha(1+\omega^2))) \qquad (3\text{-}53)$$

où $K = |G|^2 + \alpha.(1+\omega^2)$ (3-54)

L'erreur totale de reconstitution sera :

$\Delta\varepsilon = |C_x|_{Max} \cdot eps$ (3-55)

par rapport à l'erreur de reconstitution $\varepsilon[\overset{\circ}{x}]$, $\Delta\varepsilon$ doit être trés faible, elle est estimée à ce que :

$\Delta\varepsilon \leq 10\,\varepsilon[\overset{\circ}{x}]$ (3-56)

$\Rightarrow \quad |C_x|_{Max} \cdot 5.10^{-m} \leq 10 \cdot \varepsilon[x]$ (3-57)

$$m \geq -\frac{\log\left(\frac{10 \cdot \varepsilon[\overset{\circ}{x}]}{5.|C_x|_{Max}}\right)}{\log(10)} \qquad (3\text{-}58)$$

Pour différentes valeurs de N et différentes valeurs de σ_η, les tableaux suivantes donnent la valeur de m :

* Sans bruit additif

N	α	$\varepsilon[x]$	$\lvert C_x\rvert_{Max}$	m
64	5.2589E-5	1.5775E-3	245.697	4.8194
128	1.8683E-4	1.8586E-3	317.266	4.9312
256	7.2563E-4	1.9335E-3	461.608	5.0769

* Avec bruit additif

- pour N = 128

σ_η	α	$\varepsilon[x]$	$\lvert C_x\rvert_{Max}$	m
0.1	1.6	4.4549E-2	32.0530	2.556
0.01	0.2	7.7652E-3	32.4100	3.3195
0.001	0.015	2.9985E-3	49.3100	3.9150

- **pour N = 256**

σ_η	α	$\varepsilon[x]$	$\lvert C_x \rvert_{Max}$	m
0.1	3.5	3.4273E-2	43.2496	2.8
0.01	0.4	6.9015E-3	42.9300	3.4928
0.001	0.013	2.9052E-3	137.945	4.3755

On remarque que pour σ_η diminuant, le paramètre m augmente et cela à cause de la variation des données qui devienne trés petite, donc une exactitude plus grande pour la représentation de ses données. On voit aussi que la valeur minimale de m est 5, donc eps = 510^{-5}, ce qui correspond comme nombre de bit minimale B:

$$eps = 2^{-B+1} \quad ======> \quad \mathbf{B = 15} \qquad (3\text{-}59)$$

En conclusion, il faut au minimum 15 bits de données pour représenter les données afin d'assurer une meilleur exactitude de reconstitution. Pour le DSP56001 , il a 24 bits de données ce qui assure une marge suffisante pour la représentation des données de mesure.

III-2-6 Résultats de l'implantation de l'algorithme dans le processeur numérique de signaux

Pour l'implantation de l'algorithme de déconvolution dans le DSP56001, la relation (3-35) a été utilisée. Cependant, l'algorithme de détermination du paramètre de régularisation α ne sera pas implanté, mais selon les résultats de simulation on introduit la valeur de α dans l'algorithme de (3-35), et par une transformation de Fourier inverse on détermine $x(\lambda)$.

Les résultats pratiques, obtenus pour les mêmes données que dans la simulation sont présentés aux figures 3.23-a, 3.23-b, 3.24-a, 3.24-b,

3.25-a et 3.25-b.

ε_{moy} désigne l'erreur moyenne de reconstitution :

$$\varepsilon_{moy} = \hat{x}(\lambda) - x(\lambda) \qquad (3\text{-}60)$$

Ces résultats confirment l'efficacité de la méthode de déconvolution utilisée pour reconstituer le spectre réel des données .

L'exactitude de reconstitution dépend de la crédibilité de l'information a priori du bruits qui affectent les données de mesure et du nombre de points de discrétisation N.

L'erreur de reconstitution est trés sensible à la valeur de α pour une variance de bruit estimée, comme le montrent les figures 3.26, 3.27 et 3.28 et les tableaux 3.2-I, 3.2-II et 3.2-III (résultats pratiques). La différence entre les résultats de simulations et pratiques est due au type d'arithmétique, à points flottants pour la simulation (PCMATLAB- 52 bits) et à points fixes pour la pratique (DSP56001-24 bits), utilisé pour le traitement des données.

En conclusion, l'application du processeur numérique de signal pour la reconstitution des signaux, présente l'intéret en spectroscopie de permettre la séparation des raies du spectre mesuré et améliorer la résolution des données spectrométriques. L'introduction des informations a priori sur le bruit dans la solution finale conduit à une solution stable proche du spectre réel. Cette application ouvre la voie à l'utilisation des processeurs numériques pour la résolution du problème de déconvolution, rencontré dans de nombreux domaines tel que le radioastronomie, géophysique, colorimétrie, etc.

Les programmes de simulations et d'expérimentations se trouvent à l'annexe A-3.

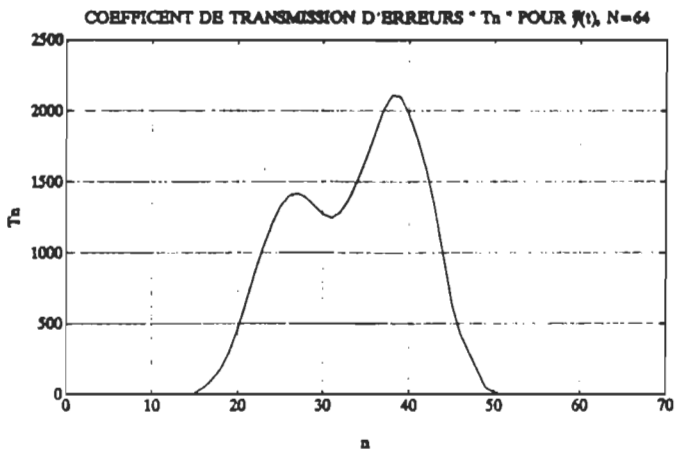


Fig.3.21-a

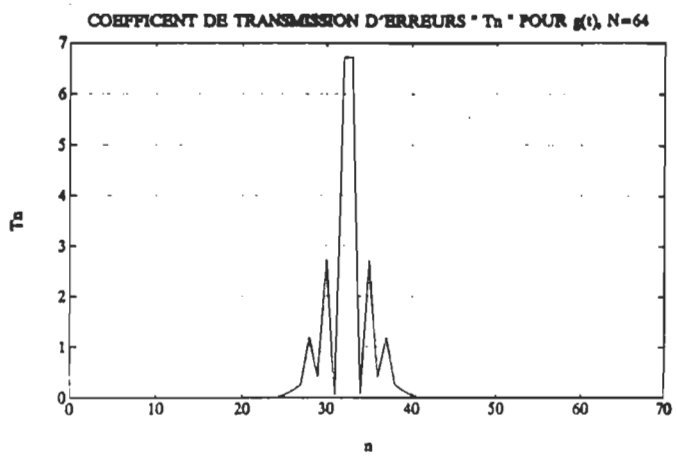


Fig.3.22-a

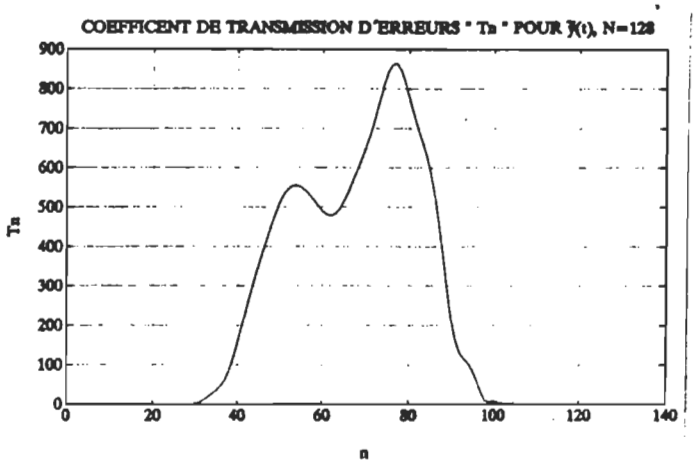


Fig.3.21-b

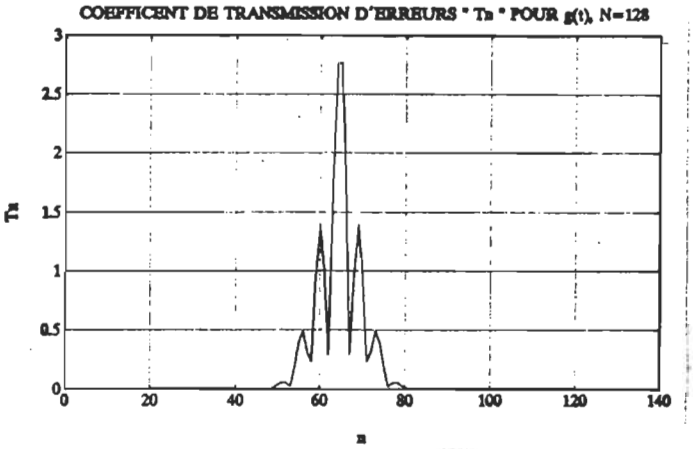


Fig.3.22-b

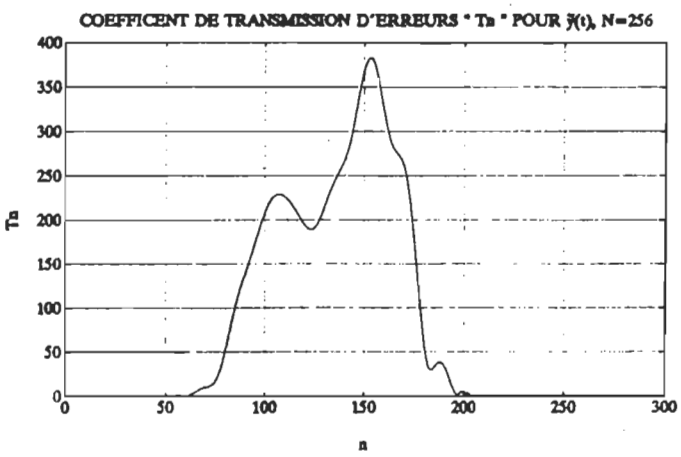


Fig.3.21-c

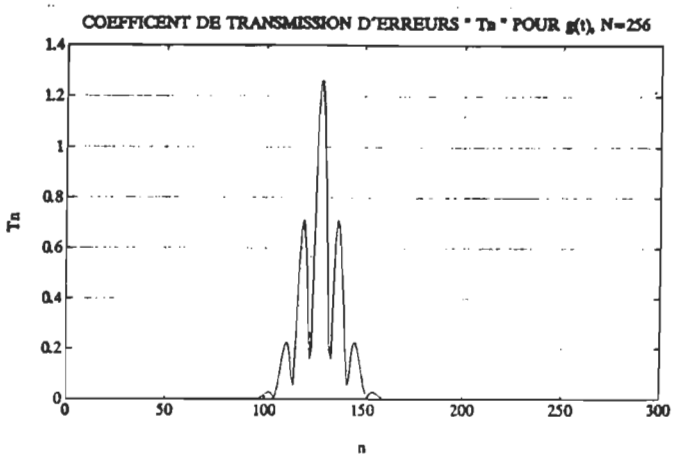


Fig.3.22-c

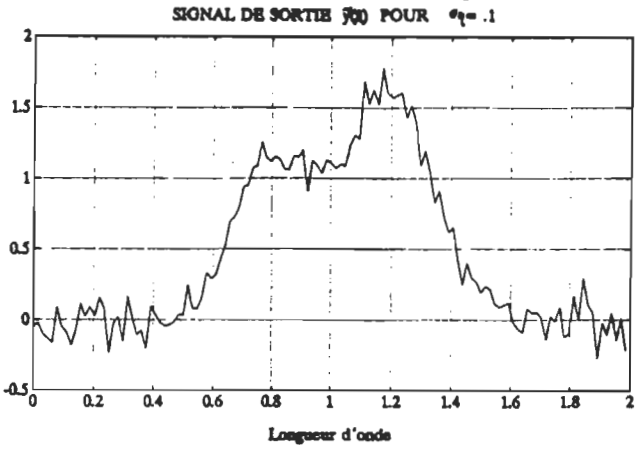


Fig.3.23-a

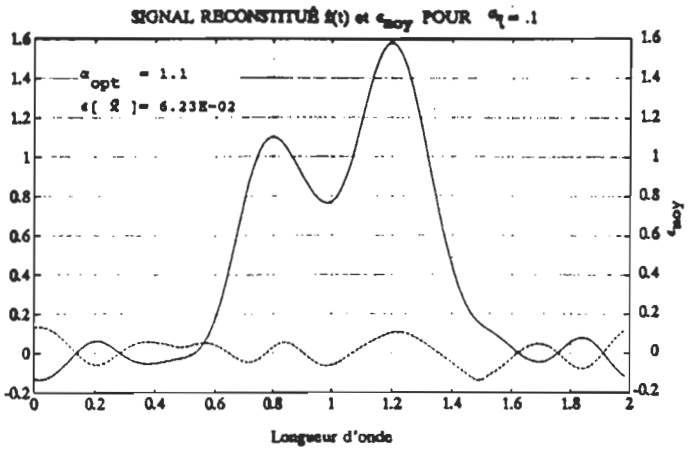


Fig.3.23-b

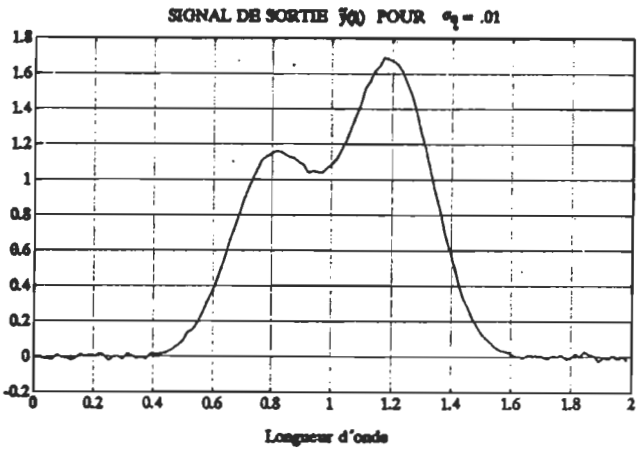


Fig.3.24-a

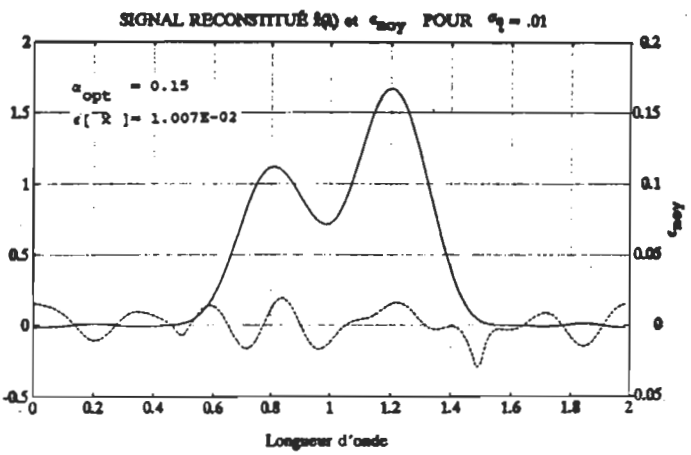


Fig.3.24-b

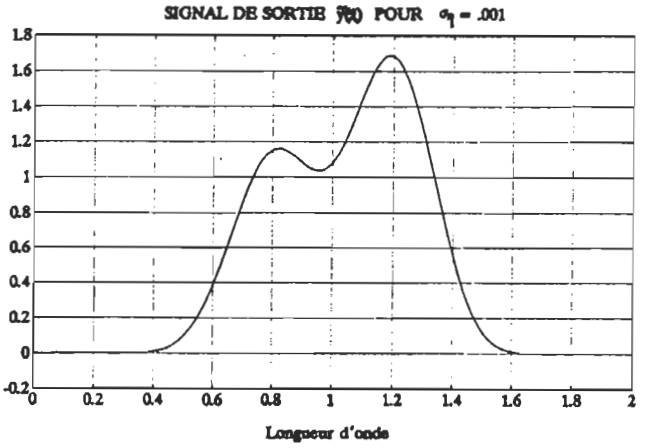


Fig.3.25-a

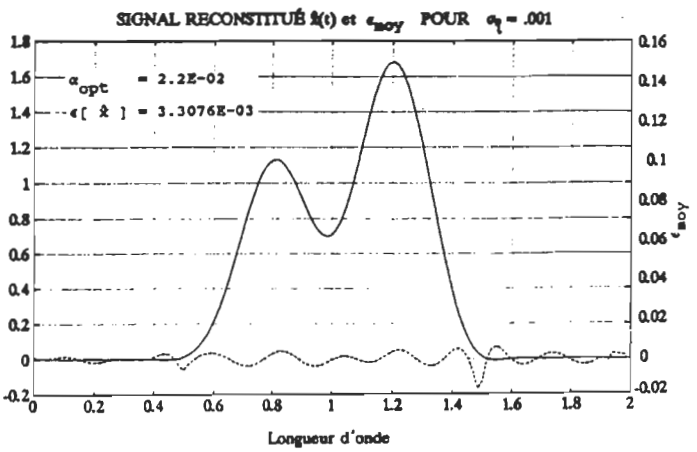


Fig.3.25-b

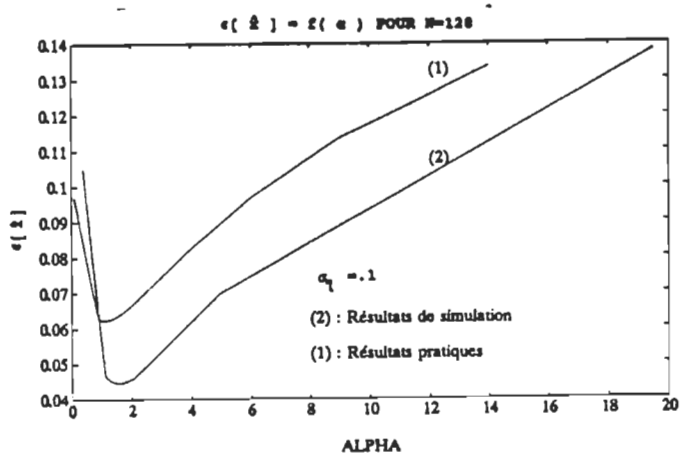


Fig.3.26

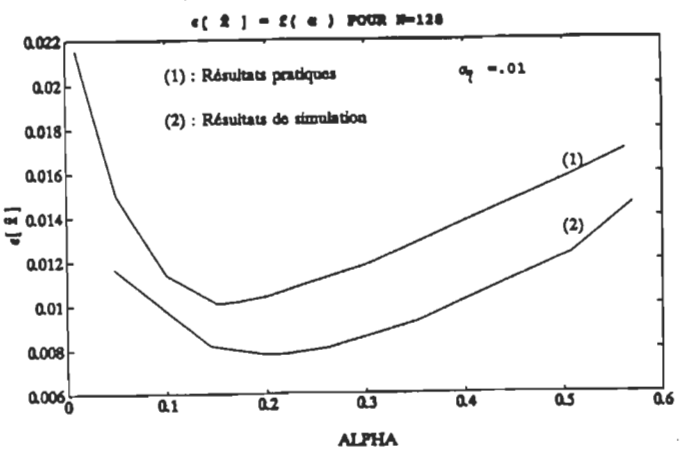


Fig.3.27

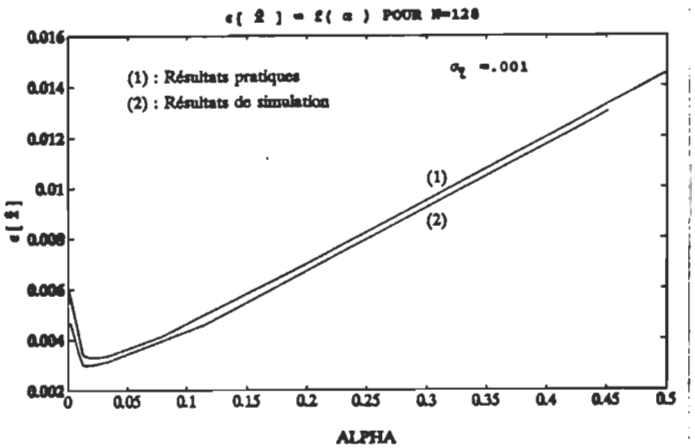


Fig.3.28

TABLEAU 3.2-I

(Résultats d'expérimentations)

avec σ_η = .1 et N = 128

α	ϵ [$\hat{x}$]
.05	11.9840E-02
.1	9.6910E-02
.9	6.2709E-02
1.0	6.2398E-02
1.1	6.2303E-02
1.2	6.2380E-02
1.4	6.2940E-02
1.7	6.4474E-02
2.0	6.6496E-02
4.0	8.2585E-02
6.0	9.6770E-02
9.0	11.3480E-02
14.0	13.3560E-02

TABLEAU 3.2-II

(Résultats d'expérimentations)

avec σ_η = .01 et N = 128

α	ϵ [$\hat{x}$]
1.00E-04	13.6743E-02
1.00E-03	5.7470E-02
1.00E-02	2.1561E-02
1.00E-01	1.0358E-02
1.50E-01	1.0070E-02
1.70E-01	1.0137E-02
2.00E-01	1.0358E-02
3.00E-01	1.1740E-02
5.00E-01	1.5700E-02
5.63E-01	1.7020E-02

TABLEAU 3.2-III

(Résultats d'expérimentations)

avec σ_η = .001 et N = 128

α	ϵ [$\hat{x}$]
1.100E-03	5.9330E-03
1.286E-02	3.3916E-03
1.559E-02	3.3420E-03
1.802E-02	3.3195E-03
1.900E-02	3.3109E-03
2.000E-02	3.3095E-03
2.200E-02	3.3076E-03
2.500E-02	3.3138E-03
2.800E-02	3.3265E-03
3.500E-02	3.3900E-03
8.000E-02	4.1500E-03
2.000E-01	6.9740E-03
5.000E-01	14.5270E-03

III-3 ÉTALONNAGE STATIQUE D'UN SYSTEME DE MESURE ÉLECTRIQUE EN UTILISANT LES FONCTIONS SPLINE

III-3-1 Formulation du problème

Soit le système de mesure électronique pour l'analyse ultrasonore des solutions défini par le schéma suivant [BARWICZ et al '90] (fig.3.29):

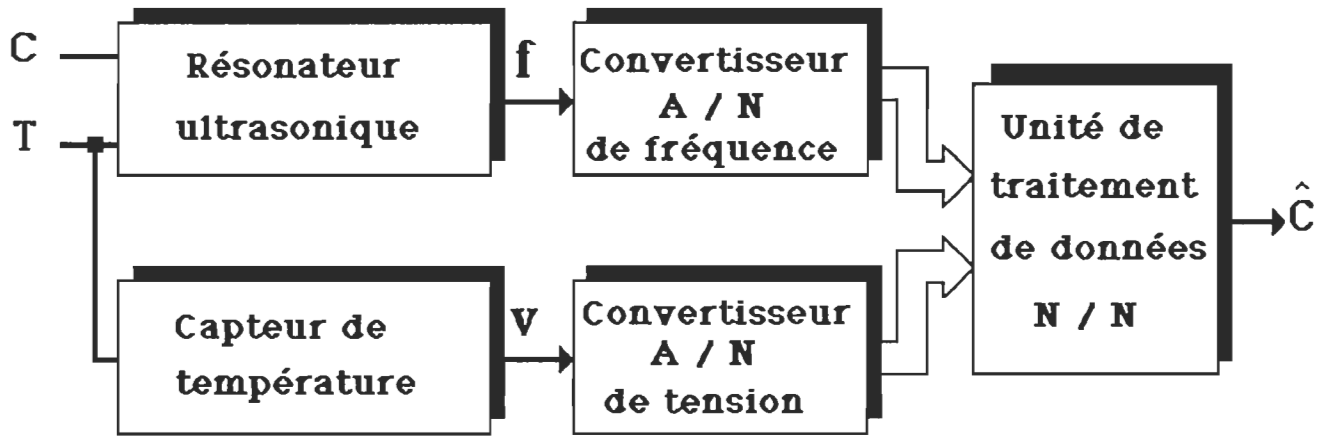


Fig.3.29 Structure du système de mesure pour l'analyse ultrsonore des solutions

Les variables qui caractérisent ce système sont :

C [%] : concentration de la solution à mesurer;

f [kHz] : fréquence de résonance;

T [°C] : température de la solution.

L'analyse ultrasonore des solutions est basée sur la relation existant entre les paramètres de la solution analysée et la vitesse et l'atténuation du son dans cette solution [BARWICZ & DION '88].

Vu la structure générale d'un système de mesure [BARWICZ '85], les exigences des blocs fonctionnels nécessaires résultent de l'exactitude et de la vitesse des mesures électroniques à faire, pour assurer l'exactitude désirée de la mesure de la concentration (dans notre cas). Les études

faites [BARWICZ & DION '88], ont permis l'acquisition des données en vu de l'établissement de la relation entre la concentration de la solution et les grandeurs mesurées, à savoir la fréquence et la température :

$$C = g (f ; T) \qquad (3\text{-}61)$$

Dans le cas d'une solution d'eau avec l'éthanol, une approximation de la relation (3-54) est donnée par l'expression suivante [BARWICZ & DION'88]:

$$C = \arg_c \{ f = K_0 (T) + K_1 (T) . C + K_2 (T) . C^2 \} \qquad (3\text{-}62)$$

où $K_0 (T) = 493.6 + 1.66\ T - .014\ T^2$ (3-63)

$K_1 (T) = 2.24 - .064\ T - .00095\ T^2$ (3-64)

$K_2 (T) = .014 - .00005\ T - .0024\ T^2$ (3-65)

L'étalonnage du système de mesure est basé sur les résultats de mesure de la concentration du méthanol dans un ensemble de solutions chimiques de référence Ci (i = 1,2,...,NC). Chaque solution de référence est expérimentée pour différentes valeurs de la température Tj (j= 1,2,...,NT). La fréquence correspondante pour chaque paire (Ci ; Tj) est mesurée aussi à l'aide d'un fréquencemètre. Toutes ces mesures sont sujet à des erreurs, ce qui engendre une erreur, parfois énorme, sur la valeur de la concentration. Il faut donc étalonner le système en essayant d'approximer le maximum possible les caractéristiques statiques du système de mesure, afin de minimiser les erreurs de mesure de C, en se basant sur le modèle des données suivants :

$$\{ Ci ; \{ Tij , fij \text{ pour } j= 1,2,\ldots,NT \} \text{ pour } i =1,2,\ldots, NC \}$$

III-3-2 Utilisation des fonctions spline pour l'étalonnage du système de mesure

En général, les fonctions sont les outils mathématiques de base pour la description et l'analyse des processus physiques. Pendant que dans

certains cas ces fonctions sont connues explicitement, il est nécessaire parfois de construire une approximation de ces fonctions en se basant sur les informations limites du processus en étude. En effet, il faut établir une approximation à une fonction inconnue, en se basant sur des séquences de données de mesure du système. Ses données sont sujet à des erreurs, et parfois ne donnent pas une approximation unique.

Une fonction dite " spline ", est une fonction formée de morceaux de polynômes qui se raccordent ainsi que certaines de leurs dérivées aux points de jonction [SCHUMAKER '81].

Dans notre cas deux types de fonctions spline, furent analysés, pour étalonner le système de mesure :

- les fonctions spline cubique;
- les fonctions spline parabolique.

La séquence des données à interpoler est définie par la fonction suivante:

$$Y_n = Y (X_n) \qquad n = 1,2,\ldots,N \tag{3-66}$$

où N est le nombre de point de mesure et (Xn;Yn) les données de mesure

III-3-2-1 Étude du spline cubique

En mathématique cette " spline " sera un ensemble de cubiques. Entre deux points donnés on utilise un polynôme cubique. Au point de jonction de deux cubiques les dérivées premières et les dérivées secondes doivent coincider afin d'obtenir les mêmes pentes et les mêmes courbures.

Pour l'intervalle (X_n ; X_{n+1}), le spline cubique est donné par la relation suivante :

$$Y_n = A_n.(x-X_n)^3 + B_n.(x-X_n)^2 + C_n.(x-X_n) + D_n \tag{3-67}$$

où A_n, B_n, C_n et D_n sont les coefficients du spline définis pour chaque noeud (Xn ; Yn) de la fonction à interpoler (relation (3-59)) (fig 3.30).

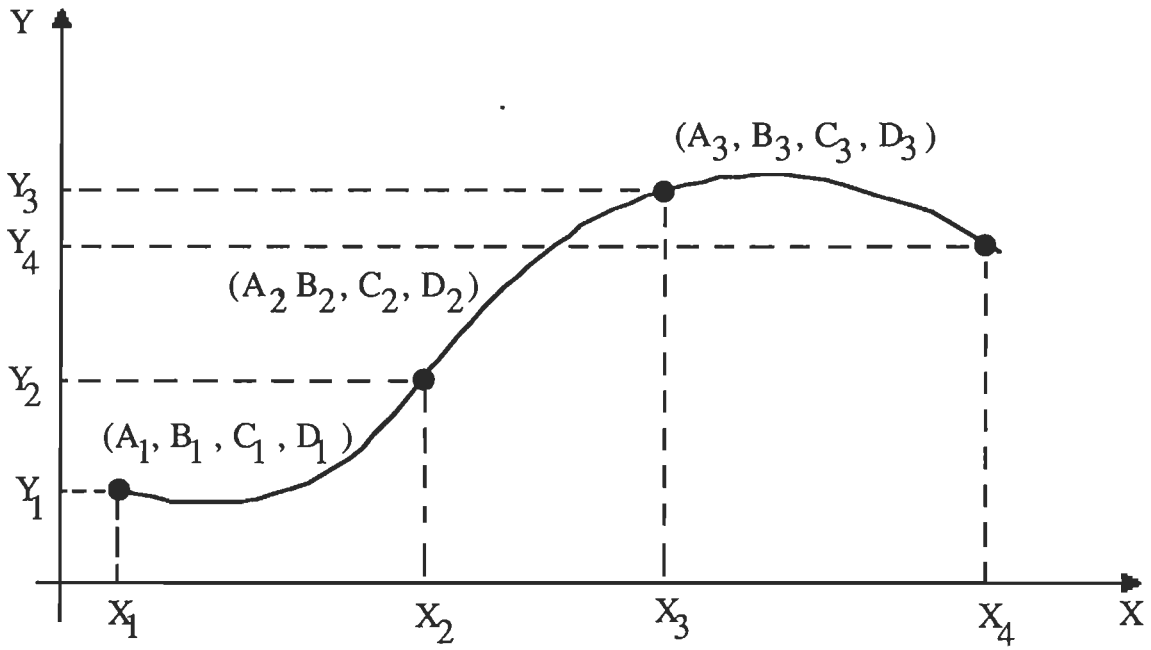


Fig.3.30 Exemple d'interpolation avec fonctions spline cubique

au point X_n, on a à partir de (3-67):

$$Y_n = D_n \tag{3-68}$$

au point X_{n+1}, on a:

$$Y_{n+1} = A_n.(h_n)^3 + B_n.(h_n)^2 + C_n.(h_n) + D_n = D_{n+1} \tag{3-69}$$

où $h_n = X_{n+1} - X_n$ (3-70)

Comme on l'a mentionné précédemment, il faut qu'aux points de jonction la fonction spline doit être continue. Les conditions suivantes doivent être, donc, établies pour l'intervalle $(X_n;X_{n+1})$ avec n=1,2,. . . , N :

$$Y'_n = \frac{dY_n}{dx}\Big|_{X_{n+1}} = Y'_{n+1} = \frac{dY_{n+1}}{dx}\Big|_{X_{n+1}} \tag{3-71}$$

$$Y''_n = \frac{d^2Y_n}{dx^2}\Big|_{X_{n+1}} = Y''_{n+1} = \frac{d^2Y_{n+1}}{dx^2}\Big|_{X_{n+1}} \tag{3-72}$$

(3-71) <====> $3.A_n.(h_n)^2 + 2.B_n.(h_n) + C_n = C_{n+1}$ (3-73)

(3-72) <====> $6.A_n.(h_n) + 2.B_n = 2.B_{n+1}$ (3-74)

III-3-2-1-1 Détermination des coefficients du spline cubique

a- Méthode générale

Cette méthode consiste à déterminer les coefficients An, Bn, Cn et Dn pour chaque noeud (X_n , Y_n) pour n=1,2,. . . , N en utilisant comme conditions initiales la connaissance de la première dérivée de la fonction pour les limites $x=X_1$ et $x=X_N$. On pose les paramètres suivants :

$u_n = Y_{n+1} - Y_n$ (3-75)

$S_n = u_n/h_n$ (3-76)

$A_n' = A_n.(h_n)^2$ pour n=1,2,. . . ,N-1 (3-77)

$B_n' = B_n.(h_n)$ pour n=1,2,. . . ,N-1 (3-78)

les relations précédentes seront :

$A_n' + B_n' + C_n = S_n$ (3-79)

$3.A_n' + 2.B_n' + C_n = C_{n+1}$ (3-80)

$3.A_n' + B_n' = B_{n+1}' . h_n/h_{n+1}$ (3-81)

en éliminant A_n' des trois dernières équations, on aura :

$C_{n+1} = 3.S_n - B_n' - 2.C_n$ (3-82)

$B_{n+1}' = K_n.(3.S_n - 2.B_n' - 3.C_n)$ (3-83)

où $K_n = h_n/h_{n+1}$ (3-84)

(3-82) <====> $B_n' = 3.S_n - C_{n+1} - 2.C_n$ (3-85)

<====> $B_{n+1}' = 3.S_{n+1} - C_{n+2} - 2.C_{n+1}$ (3-86)

en égalisant (3-86) avec (3-83) on obtient :

$C_n = 3.(S_{n-1} + K_{n-2}.S_{n-2}) - 2.(1+K_{n-2}).C_{n-1} - K_{n-2}.C_{n-2}$ (3-87)

on pose $R_n = 3.(S_{n-1} + K_{n-2}.S_{n-2})$ (3-88)

$P_n = -2.(1+K_{n-2})$ (3-89)

d'où $C_n = P_n.C_{n-1} - K_{n-2}.C_{n-2} + R_n$ (3-90)

À partir de cette relation, on détermine une relation récurrente entre C_n et C_2 tel que :

$$C_n = \beta(n).C_2 + \delta(n) \qquad \text{pour } n=3,\ldots,N-1 \tag{3-91}$$

où $\beta(n) = P_n.\beta(n-1) - K_{n-2}.\beta(n-2)$ (3-92)

$$\delta(n) = P_n.\delta(n-1) - K_{n-2}.\delta(n-2) + R_n \tag{3-93}$$

Pour démontrer cette relation, on procède par récurrence :

on a $C_n = P_n.C_{n-1} - K_{n-2}.C_{n-2} + R_n$

on choisit les conditions initiales suivantes :

$\beta(1) = 0$ $\qquad\qquad$ $\delta(1) = C_1$

$\beta(2) = 1$ $\qquad\qquad$ $\delta(2) = 0$

pour **n =3**

$$C_3 = P_3.C_2 - K_1.C_1 + R_3 \tag{3-94}$$

$$C_3 = (P_3.\beta(2) - K_1.\beta(1)).C_2 + (P_3.\delta(2) - K_1.\delta(1) + R_3)$$

===> $C_3 = \beta(3).C_2 + \delta(3)$

la relation (3-91) est vraie pour n=3. On va supposer qu'elle est encore vraie jusqu'a (n-1) et on essaie de vérifier la relation pour n. On a à partir de (3-91)

$$C_{n-1} = \beta(n-1).C_2 + \delta(n-1) \tag{3-95}$$

$$C_{n-2} = \beta(n-2).C_2 + \delta(n-2) \tag{3-96}$$

En remplaçant ces deux relations dans la relation (3-90), on aura :

$$C_n = P_n.(\beta(n-1).C_2 + \delta(n-1)) - K_{n-2}.(\beta(n-2).C_2 + \delta(n-2)) + R_n$$

$\Longrightarrow C_n = (P_n.\beta(n-1) - K_{n-2}.\beta(n-2)).C_2 + (P_n.\delta(n-1) - K_{n-2}.\delta(n-2) + R_n)$

$\Longrightarrow C_n = \beta(n).C_2 + \delta(n)$

la relation est, donc, vraie pour n.

La valeur de C_2 peut être déterminer à partir de cette dernière realtion pour le cas où n=N

$$C_N = \beta(N).C_2 + \delta(N) \tag{3-97}$$

or $C_N = dY_N/dx$ ===> $C_2 = (C_N - \delta(N))/\beta(N)$

Dans ce cas, on peut déterminer le coefficient C_n pour n=3,. . . ,N-1 et de cela on détermine les autres coefficients A_n, B_n, et D_n à partir des relations :

$Dn = Yn$ pour n=1,. . . ,N

$C_{n+1} = \beta(n+1).C_2 + \delta(n+1)$ pour n=1,. . . ,N-1 (3-98)

$B_n = (3.S_n - C_{n+1} - 2.C_n)/h_n$ pour n=1,2,. . . ,N-1 (3-99)

$A_n = (C_{n+1} + C_n - 2.S_n)/(h_n)^2$ pour n=1,2,. . . ,N-1 (3-100)

En ce qui concerne la détermination des valeurs des premières dérivées pour les valeurs limites n=1 et n=N ($C_1 = dY_1/dx$ et $C_N = dY_N/dx$), on a utilisé la formule de Lagrange [BLUM'72] pour approximer ces valeurs numériquement, à savoir :

$$Y(x) = \sum_{i=0}^{N} Y_i \; L_i(x) \qquad (3\text{-}101)$$

avec

$$L_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{N} \left(\frac{x - x_j}{x_i - x_j} \right) \qquad (3\text{-}102)$$

et cela pour les quatres premiers points X_1, X_2, X_3 et X_4 pour C_1 et X_N, X_{N-1}, X_{N-2} et X_{N-3} pour la détermination de C_N.

$$C_1 = \frac{dY}{dx}\Big|_{X_1}$$

$$C_1 = Y_1 . \frac{dL_1}{dx}\Big|_{X_1} + Y_2 . \frac{dL_2}{dx}\Big|_{X_1} + Y_3 . \frac{dL_3}{dx}\Big|_{X_1} + Y_4 . \frac{dL_4}{dx}\Big|_{X_1} \qquad (3\text{-}103)$$

et de la même façon, on détermine C_N pour n=N.

b- Méthode avec optimisation du spline

Cette méthode consiste à la détermination des coefficients du spline

cubique avec optimisation du spline, C'est-à-dire en minimisant la norme de sa troisième dérivée dans l'espace Hilbert (L2) définie par :

$$J = \frac{1}{72} \left\| \frac{d^3Y}{dx^3} \right\|_{L2}^2 = \frac{1}{72} \int_{X_1}^{X_N} \left| \frac{d^3Y}{dx^3} \right|^2 dx \qquad (3\text{-}104)$$

$$J = \frac{1}{2} \sum_{n=1}^{N-1} A_n^2 \cdot h_n = \frac{1}{2} \sum_{n=1}^{N-1} \frac{(A'_n)^2}{h_n^3}$$

Pour cela, on détermine une relation récurrente entre A'_n, B'_1 et C_1 de la forme :

$$A_n' = \alpha(n) + \beta(n).B_1' + \delta(n).C_1 \qquad \text{pour } n=3,\ldots,N\text{-}1 \qquad (3\text{-}105)$$

avec $\alpha(n) = P_n.\alpha(n\text{-}1) - Q_n.\alpha(n\text{-}2) + R_n$ (3-106)

$\beta(n) = P_n.\beta(n\text{-}1) - Q_n.\beta(n\text{-}2)$ (3-107)

$\delta(n) = P_n.\delta(n\text{-}1) - Q_n.\delta(n\text{-}2)$ (3-108)

$Q_n = (h_n + h_{n-1})/(h_{n-1} + h_{n-2})$ (3-109)

$P_n = Q_n - 2 + 3.h_n/h_{n-1}$ (3-110)

$R_n = S_n - S_{n-1}.(1+Q_n) + S_{n-2}.Q_n$ (3-111)

Pour démontrer ces relations, on procède par récurrence. On a :

$A_1' = S_1 - B_1' - C_1$ ======> $\alpha(1) = S1$; $\beta(1) = -1$; $\delta(1) = -1$

de même $A_2' = S_2 - B_2' - C_2$ avec

$C_2 = 3.S_1 - B_1' - 2.C_1$ et $B_2' = (3.S_1 - 2.B_1' - 3.C_1).h_2/h_1$ ===>

$\alpha(2) = S2 - 3.(1+h_2/h_1)$; $\beta(2) = 1+2.h_2/h_1$; $\delta(2) = 2 + 3.h_2/h_1$

vérifions pour le cas **n=3**

$A_3' = S_3 - B_3' - C_3$

et en remplaçant B_3' et C_3 par leur expression en fonction de B_1' et C_1 on aboutit à la relation suivante:

$A_3' = \alpha(3) + \beta(3).B_1' + \delta(3).C_1$

où $\alpha(3) = S_3 - S_2 - (2+3.h_3/h_2).\alpha(2) - 3.S_1.(h_2/h_1 + h_3/h_2)$

$\beta(3) = -(2+3.h_3/h_2).\beta(2) + 2.(1+h_2/h_1).(h_2+h_3)/(h_2+h_1)$

$\delta(3) = -(2+3.h_3/h_2).\delta(2) + 3.(h_2+h_3)/h_1$

aprés quelques reformulations des équations on peut trouver facilement les expressions suivantes :

$\alpha(3) = P_3.\alpha(2) - Q_3.\alpha(1) + R3$

$\beta(3) = P_3.\beta(2) - Q_3.\beta(1)$

$\delta(3) = P_3.\delta(2) - Q_3.\delta(1)$

avec $Q_3 = (h_2+h_3)/(h_2+h_1)$

$P_3 = Q_3 - 2 + 3.h_2/h_1$

$R_3 = S_3 - (1+Q_3).S_2 + Q_3.S_1$

La relation (3-105) est vraie pour n=3, et on suppose qu'elle est aussi vraie jusqu'a (n-1). On essayede vérifier pour n. À partir des relations (3-79), (3-80) et (3-81) on peut déduire la relation suivante :

$$B'_{n+1} = \frac{h_{n+1}}{h_{n+1} + h_n} [\ S_{n+1} - S_n + A'_n - A'_{n+1}\ ] \qquad (3\text{-}112)$$

d'autre part, (3-80) ===> $3.A_n' + 2.B_n' + C_n = C_{n+1}$ ====>

$C_{n+1} = 2.A_n' + B_n' + S_n$ ====>

$$C_{n+1} = 2.A'_n + S_n + \frac{h_n}{h_n + h_{n-1}} [\ S_n - S_{n-1} + A'_{n-1} - A'_n\ ] \qquad (3\text{-}113)$$

(3-79) ====> $A_n' = S_n - B_n' - C_n$

et en remplaçant B_n' et C_n par leurs expressions (3-112) et (3-113), on peut trouver la relation suivante, aprés quelques reformulations :

$$A'_n = R_n + P_n . A'_{n-1} - Q_n . A'_{n-2} \qquad (3\text{-}114)$$

d'autre part, on avait dit que la relation (3-105) est vraie jusqu'a (n-1), donc on pourra écrire :

$A'_{n-1} = \alpha(n-1) + \beta(n-1).B'_1 + \delta(n-1).C_1$

$A'_{n-2} = \alpha(n-2) + \beta(n-2).B'_1 + \delta(n-2).C_1$

en remplaçant ces deux dernières équations dans (3-114), on trouve :

$$A_n' = R_n + P_n.\alpha(n-1) - Q_n.\alpha(n-2) + [\, P_n.\beta(n-1) - Q_n.\beta(n-2)].B_1' + [\, P_n.\delta(n-1) - Q_n.\delta(n-2)\,].C_1$$

$$\Longrightarrow \quad A_n' = \alpha(n) + \beta(n).B_1' + \delta(n).C_1 \qquad \text{pour } n=3,\ldots,N-1$$

donc la relation est vraie pour tout n.

Donc, pour que J soit minimum il faut que :

$$\frac{\partial J}{\partial B_1'} = 0 \qquad \text{et} \qquad \frac{\partial J}{\partial C_1} = 0$$

$$\frac{\partial J}{\partial B'_1} = 0 = \sum_{n=1}^{N-1} \frac{A'_n}{h_n^3} \cdot \frac{\partial A'_n}{\partial B'_1} \tag{3-115}$$

et aussi :

$$\frac{\partial J}{\partial C_1} = \sum_{n=1}^{N-1} \frac{A'_n}{h_n^3} \cdot \frac{\partial A'_n}{\partial C_1} \tag{3-116}$$

$$\frac{\partial A'_n}{\partial B'_1} = \beta(n) \qquad \text{et} \qquad \frac{\partial J}{\partial C_1} = \delta(n) \tag{3-117}$$

(3-115) $\Longrightarrow$ $\Sigma 11 \,.\, B'_1 + \Sigma 12 \,.\, C_1 = \Sigma 1$

(3-116) $\Longrightarrow$ $\Sigma 12 \,.\, B'_1 + \Sigma 22 \,.\, C_1 = \Sigma 2$

où

$$\Sigma 11 = \sum_{n=1}^{N-1} d_n \,.\, (\beta(n))^2 \quad ; \quad \Sigma 12 = \sum_{n=1}^{N-1} d_n \,.\, \beta(n) \,.\, \delta(n) \tag{3-118}$$

$$\Sigma 22 = \sum_{n=1}^{N-1} d_n \,.\, (\delta(n))^2 \quad ; \quad \Sigma 1 = -\sum_{n=1}^{N-1} d_n \,.\, \alpha(n) \,.\, \beta(n) \tag{3-119}$$

$$\Sigma 2 = -\sum_{n=1}^{N-1} d_n \,.\, \alpha(n) \,.\, \delta(n) \quad ; \quad d_n = \frac{1}{h_n^3} \tag{3-120}$$

On aura, enfin, un système d'équation à partir duquel on peut déterminer B_1' et C_1.

$$B'_1 = \frac{\sum 1 \cdot \sum 22 \; - \; \sum 2 \cdot \sum 12}{\sum 22 \cdot \sum 11 \; - \; \sum 12 \cdot \sum 12} \qquad (3\text{-}121)$$

$$C_1 = \frac{\sum 1 \; - \; \sum 11 \cdot B'_1}{\sum 12} \qquad (3\text{-}122)$$

Les coefficients A_n' seront donc déterminés en utilisant les deux dernières relations:

$$A_n = [\, \alpha(n) + \beta(n).B_1' + \delta(n).C_1 \,] \,/(h_n)^2 \quad \text{pour } n=3,\ldots,N\text{-}1 \qquad (3\text{-}123)$$

Pour les coefficients B_n et C_n on a :

$$C_1 = S_1 - B_1' - A_1'$$

(3-80) ====> $C_{n+1} = C_n + 2.B_n' + 3.A_n'$ pour n=1,2,. . . ,N-1

(3-112) ====>

$$B'_n = \frac{h_n}{h_n + h_{n-1}} [\, S_n - S_{n-1} + A'_{n-1} - A'_n \,] \qquad =====>$$

$$B_2' = [S_2 - S_1 + A_1' - A_2'].h_2/(h_2 + h_1)$$

(3-81) ===> $B_2' = (3.A_1' + B_1').h_2/h_1$

en égalisant les deux équations, on peut déduire :

$$B'_1 = \frac{1}{1+z} [\, S_2 - S_1 - A'_2 - A'_1.(2 + 3.z) \,] \qquad (3\text{-}124)$$

où $z = h_2/h_1$ et à partir de la relation (3-81) :

$$B'_{n+1} = (3.\, A'_n + B'_n) \cdot \frac{h_{n+1}}{h_n} \qquad \text{pour } n=1,\ldots,N\text{-}1$$

et enfin pour le coefficient Dn

$$D_n = Y_n \qquad \text{pour } n=1,2,\ldots,N$$

III-3-2-2 Étude du spline parabolique

Le spline parabolique désigne un polynôme parabolique (degré 2) entre deux points donnés. Au point de jonction de deux polynômes paraboliques les dérivées premières et secondes doivent coincider afin d'obtenir les mêmes pentes et les mêmes courbures.

Pour l'intervalle (X_n ; X_{n+1}), le spline parabolique est donné par la relation suivante :

$$Y_n = A_n.(x-X_n)^2 + B_n.(x-X_n) + C_n \qquad (3\text{-}125)$$

où A_n, B_n et C_n sont les coefficients du spline définis pour chaque noeud (Xn ; Yn) de la fonction à interpoler (relation (3-66)).

On garde les mêmes notations que pour le spline parabolique concernant les relations (3-75), (3-76) et (3-70).

au point X_n, on a :

$$Y_n = C_n \qquad (3\text{-}126)$$

au point X_{n+1}, on a:

$$Y_{n+1}= A_n.(h_n)^2 + B_n.(h_n) + C_n= C_{n+1} \qquad (3\text{-}127)$$

Comme on l'a mentionné précédemment, il faut qu'aux points de jonction la fonction spline doit être continue. Les conditions établies pour le spline cubique seront les mêmes pour le spline parabolique pour l'intervalle (X_n ; X_{n+1}) avec n=1,2,. . . , N-1.

(3-127) ===> $A_n.(h_n)^2 + B_n.(h_n) + C_n= C_{n+1}$ (3-128)

(3-71) ===> $2.A_n.(h_n) + B_n = B_{n+1}$ (3-129)

III-3-2-2-1 Détermination des coefficients du spline parabolique

a- Méthode générale

On détermine les coefficients A_n, B_n et C_n pour chaque noeud

$(X_n;Y_n)$ pour n=1,...,N en utilisant comme condition initiale la connaissance de la première dérivée pour X_1. On a :

(3-128) ===> $A_n = (S_n - B_n)/h_n$

(3-129) ===> $B_{n+1} = B_n + 2.A_n/h_n$

de ces deux relations on peut déduire que :

$B_{n+1} = 2.S_n - B_n$ pour n=1,2,. . .,N-1 (3-130)

$A_n = (S_n - B_n)/h_n$ pour n=1,2,. . .,N-1 (3-131)

$C_n = Y_n$ pour n=1,2,. . .,N

Pour la détermination de B_1, qui représente la première dérivée du spline on a utilisé la formule de Lagrange [BLUM'72]

$$B_1 = \left.\frac{dY_1}{dx}\right|_{X_1} \qquad (3\text{-}132)$$

b- Méthode avec optimisation du spline

Comme on l'a fait pour le spline cubique, il s'agit de minimiser une critère donnée. Dans ce cas le coefficient B_1 sera déterminé de façon à minimiser la norme de la seconde dérivée du spline parabolique dans l'espace Hilbert.

$$J = \frac{1}{8} \| \frac{d^2Y}{dx^2} \|^2_{L2} = \frac{1}{8} \int_{X_1}^{X_N} | \frac{d^2Y}{dx^2} |^2 \, dx \qquad (3\text{-}133)$$

$$J = \frac{1}{2} \sum_{n=1}^{N-1} A_n^2 \cdot h_n = \frac{1}{2} \sum_{n=1}^{N-1} \frac{A_n^2}{h_n}$$

avec $A_n' = A_n \cdot h_n$

D'autre part, à partir de (3-128) on a :

$A_n' = S_n - B_n$ ====> $A'_{n+1} = S_{n+1} - B_{n+1}$

$$====> \quad A'_{n+1} = S_{n+1} - B_n - 2.A_n'$$

$$====> \quad A'_{n+1} = S_{n+1} - S_n - A_n' \qquad (3\text{-}134)$$

pour n=1,. . .,N-1 avec $A_1' = S_1 - B_1$

Pour cela, on détermine une relation récurrente qui relie A_n' à A_1' de la forme :

$$A_n' = (-1)^{n-1}.\,(A_1' + \beta(n)) \qquad \text{pour n=1,2,. . .,N-1} \qquad (3\text{-}135)$$

avec $\beta(n+1) = \beta(n) + (-1)^n.(S_{n+1} - S_n)$ (3-136)

Pour démontrer cette relation, on procède par récurrence.

pour **n = 2**

$$A_2' = S_2 - S_1 - A_1'$$

$$= (-1).(S_1 - S_2 + A_1')$$

$$= (-1).(A_1' + \beta(2))$$

La relation (3-136) est, donc, vraie pour n=2. Supposant qu'elle est aussi vraie jusqu'a (n-1) et essayons de la vérifier pour n. On a :

$$A'_{n-1} = (-1)^{n-2}.(A_1' + \beta(n-1)) \qquad (3\text{-}137)$$

d'autre part, de la relation (3-134) :

$$A'_n = S_n - S_{n-1} - A'_{n-1}$$

$$= S_n - S_{n-1} - (-1)^{n-2}.(A_1' + \beta(n-1))$$

$$A'_n = S_n - S_{n-1} + (-1)^{n-1}.(A_1' + \beta(n-1)) \qquad (3\text{-}138)$$

de plus à partir de (3-137), on peut déduire que :

$$S_n - S_{n-1} = (-1)^{n-1}.(\beta(n) - \beta(n-1)) \qquad (3\text{-}139)$$

$$===> \quad A'_n = (-1)^{n-1}.(\beta(n) - \beta(n-1)) + (-1)^{n-1}.(A_1' + \beta(n-1))$$

$$===> \quad A'_n = (-1)^{n-1}.(\beta(n) - \beta(n-1) + A_1' + \beta(n-1)) = (-1)^{n-1}.(A_1' + \beta(n))$$

===> la relation est, donc, vraie pour n.

Pour que J soit minimum, il faut que

$$\partial J / \partial A_1' = 0 \qquad (3\text{-}140)$$

En appliquant la relation (3-140) on aboutit :

$$\frac{\partial J}{\partial A'_1} = \sum_{n=1}^{N-1} \frac{A'_n}{h_n} \cdot \frac{\partial A'_n}{\partial A'_1} = \sum_{n=1}^{N-1} (-1)^{n-1} \cdot \frac{A'_1 + \beta(n)}{h_n} = 0 \qquad (3\text{-}141)$$

$$A'_1 \cdot \sum_{n=1}^{N-1} \frac{1}{h_n} + \sum_{n=1}^{N-1} \frac{\beta(n)}{h_n} = 0 \qquad (3\text{-}142)$$

$$A'_1 = - \frac{\sum_{n=1}^{N-1} \frac{\beta(n)}{h_n}}{\sum_{n=1}^{N-1} \frac{1}{h_n}} \qquad (3\text{-}143)$$

$\Longrightarrow \quad B_1' = S_1 - A_1'$

Enfin, pour n=2,. . . ,N-1 on aura les coefficients suivantes :

$A_n' = S_n - B_n \Longrightarrow \quad A_n = (S_n - B_n)/h_n \qquad (3\text{-}144)$

$B_n = B_{n-1} + 2.A_{n-1}/h_{n-1} \qquad (3\text{-}145)$

$C_n = Y_n \qquad (3\text{-}146)$

III-3-2-3 Procédure d'étalonnage avec les fonctions spline

La procédure d'interpolation par les fonctions spline, pour l'étalonnage du système de mesure, est la suivante :

1º Pour chaque référence de concentration Ci (i=1,2,...,NC) on associe une fonction spline :

$$f = S(T , p_i) \qquad (3\text{-}147)$$

interpolant la séquence des paires de données:

$$\{ \; Tij , fij \; / \; j=1,2,\ldots\ldots, NT \; \}$$

où p_i : les vecteurs de coéfficients des fonctions d'interpolations.

2º Pour chaque température mesuré, Tmes, on détermine la fréquence corresponadante en utilisant la valeur donnée par les

fonctions d'interpolations, soit f_{est} .

3º On utilise de nouveau la procédure d'interpolation pour la séquence des paires de données :

$$\{ f_{est} \ ; Ci / i = 1,2,. . . , NC \}$$

On obtient une autre famille de courbes pour différentes valeurs de Tmes.

4º À partir de ses courbes, et pour une fréquence mesurée f_{mes} on détermine la concentration correspondante en utilisant la valeur donnée par la fonction d'interpolation C_{est} .

III-3-3 Résultats de l'étalonnage du système de mesure

III-3-3-1 Résultats de simulations

La méthodologie suivante a été utilisée pour la simulation de ce système de mesure :

- pour une valeur de C et NT valeurs de T on détermine les fréquences correspondantes à partir de la relation (3-62). On répète cette expérience NC fois pour obtenir tous les courbes:

$$f = g (C \text{ (constante)} ; T) \qquad (3\text{-}148)$$

- à chaque paramètre du système, on ajoute une erreur, représentant l'ensemble des erreurs de mesure, afin d'introduire l'aspect pratique sur les résultats de simulations :

* à C on ajoute une valeur aléatoire de distribution uniforme dans l'intervalle [-ErrC +ErrC], où ErrC désigne l'erreur maximale sur la mesure de la concentration C.

$$\Delta C = k. ErrC \text{ avec } -1 \leq k \leq +1 \qquad (3\text{-}149)$$

$$Ci = Ci + \Delta Ci \qquad (3\text{-}150)$$

* à la valeur de T on ajoute une valeur aléatoire de distribution uniforme dans l'intervalle [-DevT +DevT], où DevT designe l'erreur maximale sur la mesure de la température T.

$$\Delta T = k.DevT \quad \text{avec} \quad -1 \leq k \leq +1 \qquad (3\text{-}151)$$

$$Tij = Tij + \Delta Tij \qquad (3\text{-}152)$$

* à la valeur de la fréquence on ajoute une valeur aléatoire de distribution uniforme dans l'intervalle [-Errf +Errf], où Errf désigne l'erreur maximale sur la mesure de la fréquence f.

$$\Delta f = k.Errf \quad \text{avec} \quad -1 \leq k \leq +1 \qquad (3\text{-}153)$$

$$fij = fij + \Delta fij \qquad (3\text{-}154)$$

Les résultats de simulations présentés dans les tableaux 3.3.1-I, 3.3.1-II, 3.3.1-III et 3.3.1-IV donnent les valeurs de l'erreur entre la valeur de la concentration référence et la valeur calculée à partir des procédures d'interpolation en utilisant les deux types de spline, pour différents nombre de points de mesure (NT ; NC).

La conclusion, qu'on peut tirer de ces résultats, est que les fonctions spline utlisées offrent une meilleur approximation du modèle du système de mesure, et aussi l'erreur entre les valeurs de concentration référence et estimées est relativement faible et acceptable. En plus le système étudié ne présente pas beaucoup de non linéarité, ce qui explique les bons résultats obtenus à partir du spline parabolique, car généralement le spline cubique offre, généralement, une meilleure interpolation.

Les abbréviations utilisées dans les tableaux 3.3.1-I à 3.3.1-IV ont les significations suivantes :

EINTSPL2: programme d'évaluation des paramètres du spline parabolique pour l'étalonnage du système de mesure.

EOPTSPL2: programme d'évaluation des paramètres du spline parabolique pour l'étalonnage du système de mesure avec optimisation

du spline.

EINTSPL3: programme d'évaluation des paramètres du spline cubique pour l'étalonnage du système de mesure.

EOPTSPL3: programme d'évaluation des paramètres du spline cubique pour l'étalonnage du système de mesure avec optimisation du spline.

$\| \delta C \|_2$: désigne l'erreur efficace entre les valeurs de la concentration calculées par la procédure d'interpolation et les valeurs de références

$\| \delta C \|_{max}$: désigne l'erreur maximale entre les valeurs de la concentration calculées par la procédure d'interpolation et les valeurs de références .

III-3-3-2 Résultats expérimentaux

L'exactitude de traitement constitue l'exigence principale de cette application. En effet, la grande quantité de calcul demandée, pour l'étalonnage de ce sytème de mesure, exige l'utilisation d'un processeur à grande capacité de calcul et une large plage dynamique pour représenter les données. Le DSP56001 est l'une des solutions envisagées. Mais vu que c'est un processeur à point fixe, l'utilisation des procédures à points flottants sera indispensable pour satisfaire les exigences de ce système. En plus la compagnie Motorola a fournie le cross-compilateur C de ce processeur, ce qui a façilité la tache lors de la programmation des procédures du spline, car il suffit d'écrire les programmes en langage C et à l'aide du cross-compilateur il les transforme en code machine compréhensible par le DSP56001. En effet le langage C offre la structure d'une langage évoluée tout en permettant l'accés direct à des adresses et des mémoires.

L'étalonnage de ce système de mesure a été vérifié sans présence

d'erreurs sur les variables du système, à savoir la température, la fréquence et la concentration (ErrC = 0 et DevT = 0). Les résultats expérimentaux, présentés aux tableaux 3.3.2-I, 3.3.2-II, 3.3.2-III et 3.3.2-IV, montrent l'efficacité du DSP. En effet la différence entre les résultats de simulations et expérimentaux est relativement negligeable, à cause de l'effet de la représentation des données par l'ordinateur qui est plus exacte que celle du DSP.

Les programmes utilisés, pour le spline cubique et parabolique, se trouvent à l'annexe A-4.

1°- TABLEAU 3.3.1-I

EINTSPL2		ErrC=0	ErrC=.01	ErrC = .1	
NC ; NT		DevT = 0		DevT = 0	DevT= .5
4 ; 4	$\|\| \delta C \|\|_2$	9.9459E-4	7.3000E-3	5.6300E-2	4.3938E-2
	$\|\| \delta C \|\|_{max}$	2.7249E-3	1.3800E-2	1.1729E-1	1.1490E-1
4 ; 5	$\|\| \delta C \|\|_2$	9.9551E-4	5.6509E-3	2.9385E-2	5.9649E-2
	$\|\| \delta C \|\|_{max}$	2.7249E-3	8.9295E-3	5.6526E-2	9.5857E-2
6 ; 6	$\|\| \delta C \|\|_2$	3.2990E-4	6.2657E-3	2.9984E-2	4.5877E-2
	$\|\| \delta C \|\|_{max}$	8.2227E-4	1.2484E-2	7.3099E-2	9.0855E-2
7 ; 7	$\|\| \delta C \|\|_2$	2.2864E-4	7.4035E-3	6.7144E-2	6.4757E-2
	$\|\| \delta C \|\|_{max}$	5.3684E-4	1.7177E-2	1.5376E-1	1.3444E-1

2°- TABLEAU 3.3.1-II

EOPTSPL2		ErrC=0	ErrC=.01	ErrC = .1	
NC ; NT		DevT = 0		DevT = 0	DevT= .5
4 ; 4	$\|\| \delta C \|\|_2$	2.7900E-2	2.8268E-2	6.5780E-2	6.0915E-2
	$\|\| \delta C \|\|_{max}$	6.7500E-2	6.6370E-2	1.2923E-1	1.3679E-1
4 ; 5	$\|\| \delta C \|\|_2$	2.0500E-2	2.2110E-2	5.6183E-2	6.1126E-2
	$\|\| \delta C \|\|_{max}$	4.1000E-2	4.3100E-2	1.0256E-1	1.0708E-1
6 ; 6	$\|\| \delta C \|\|_2$	6.1792E-3	8.8419E-3	4.8069E-2	5.5905E-2
	$\|\| \delta C \|\|_{max}$	1.5118E-2	2.1804E-2	9.8670E-2	1.1977E-1
7 ; 7	$\|\| \delta C \|\|_2$	1.0160E-3	6.9617E-3	4.7269E-2	6.9114E-2
	$\|\| \delta C \|\|_{max}$	4.2879E-3	1.3521E-2	9.7263E-2	1.1500E-1

3°- TABLEAU 3.3.1-III

EINTSPL3		ErrC=0	ErrC=.01	ErrC = .1	
NC ; NT		DevT = 0		DevT = 0	DevT= .5
4 ; 4	$\|\| \delta C \|\|_2$	1.9553E-3	4.0849E-3	5.6490E-2	4.1378E-2
	$\|\| \delta C \|\|_{max}$	4.7365E-3	8.1007E-3	1.1568E-1	9.5345E-2
4 ; 5	$\|\| \delta C \|\|_2$	1.9623E-3	4.5553E-3	3.7322E-2	6.1369E-2
	$\|\| \delta C \|\|_{max}$	4.7365E-3	9.0215E-3	4.4716E-2	1.0493E-1
6 ; 6	$\|\| \delta C \|\|_2$	1.9226E-4	5.8985E-3	6.7313E-2	5.9600E-2
	$\|\| \delta C \|\|_{max}$	6.3503E-4	1.0477E-2	1.0702E-1	1.0510E-1
7 ; 7	$\|\| \delta C \|\|_2$	8.5409E-5	5.0600E-3	6.0930E-2	5.2566E-2
	$\|\| \delta C \|\|_{max}$	3.2115E-4	8.6717E-3	9.0890E-2	9.9045E-2

4°- TABLEAU 3.3.1-IV

EOPTSPL3		ErrC=0	ErrC=.01	ErrC = .1	
NC ; NT		DevT = 0		DevT = 0	DevT= .5
4 ; 4	$\|\| \delta C \|\|_2$	2.4993E-3	6.0503E-3	6.5177E-2	5.7166E-2
	$\|\| \delta C \|\|_{max}$	7.9648E-3	1.0458E-2	9.2837E-2	8.6736E-2
4 ; 5	$\|\| \delta C \|\|_2$	2.5071E-3	5.5586E-3	5.5828E-2	5.3209E-2
	$\|\| \delta C \|\|_{max}$	7.9648E-3	1.0722E-2	9.1699E-2	9.5502E-2
6 ; 6	$\|\| \delta C \|\|_2$	4.9700E-4	5.8459E-3	6.6746E-2	5.6675E-2
	$\|\| \delta C \|\|_{max}$	1.8127E-3	1.0380E-2	1.0783E-1	9.6208E-2
7 ; 7	$\|\| \delta C \|\|_2$	2.7518E-4	6.4573E-3	5.2430E-2	5.1594E-2
	$\|\| \delta C \|\|_{max}$	1.0633E-3	9.3777E-3	9.5276E-2	9.5529E-2

1°- TABLEAU 3.3.2-I: RESULTATS DU PROGRAMME " EINTSPL2 "

ErrC = 0 , DevT = 0		SIMULATION	DSP56000/1
(NC,NT) 4,4	$\| \delta C \|_2$	9.958180E-04	9.760280E-04
	$\| \delta C \|_{max}$	2.739430E-03	2.715110E-03
(NC,NT) 5,5	$\| \delta C \|_2$	5.194830E-04	5.152727E-04
	$\| \delta C \|_{max}$	1.334190E-03	1.326560E-03
(NC,NT) 6,6	$\| \delta C \|_2$	3.290880E-04	3.455980E-04
	$\| \delta C \|_{max}$	8.134840E-04	8.916850E-04
(NC,NT) 7,7	$\| \delta C \|_2$	2.291720E-04	2.432755E-04
	$\| \delta C \|_{max}$	5.521770E-04	5.731580E-04
(NC,NT) 8,8	$\| \delta C \|_2$	1.613730E-04	2.006054E-04
	$\| \delta C \|_{max}$	3.848080E-04	4.997254E-04

2°- TABLEAU 3.3.2-II : RESULTATS DU PROGRAMME " EOPTSPL2 "

ErrC = 0 , DevT = 0		SIMULATION	DSP56000/1
(NC,NT) 4,4	$\| \delta C \|_2$	2.792700E-02	1.605470E-02
	$\| \delta C \|_{max}$	6.756570E-02	6.759260E-02
(NC,NT) 5,5	$\| \delta C \|_2$	3.357800E-03	2.102830E-03
	$\| \delta C \|_{max}$	7.040000E-03	7.053375E-03
(NC,NT) 6,6	$\| \delta C \|_2$	6.177970E-03	5.576940E-03
	$\| \delta C \|_{max}$	1.509830E-02	1.507950E-02
(NC,NT) 7,7	$\| \delta C \|_2$	1.015940E-03	1.003099E-03
	$\| \delta C \|_{max}$	2.161030E-03	2.189636E-03
(NC,NT) 8,8	$\| \delta C \|_2$	2.270500E-03	1.316348E-03
	$\| \delta C \|_{max}$	5.860700E-03	5.939245E-03

3°- TABLEAU 3.3.2-III : RESULTATS DU PROGRAMME " EINTSPL3 "

ErrC = 0 , DevT = 0		SIMULATION	DSP56000/1
(NC,NT) 4,4	$\| \delta C \|_2$	1.956460E-03	1.937227E-03
	$\| \delta C \|_{max}$	4.756930E-03	4.680634E-03
(NC,NT) 5,5	$\| \delta C \|_2$	4.914230E-04	4.877424E-03
	$\| \delta C \|_{max}$	1.434800E-03	1.415729E-03
(NC,NT) 6,6	$\| \delta C \|_2$	1.927150E-04	1.885439E-04
	$\| \delta C \|_{max}$	6.229880E-04	6.570816E-04
(NC,NT) 7,7	$\| \delta C \|_2$	8.698960E-05	1.166515E-04
	$\| \delta C \|_{max}$	3.364090E-04	4.730225E-04
(NC,NT) 8,8	$\| \delta C \|_2$	5.158970E-05	1.164177E-04
	$\| \delta C \|_{max}$	2.422330E-04	8.010864E-04

4°- TABLEAU 3.3.2-IV : RESULTATS DU PROGRAMME " EOPTSPL3 "

ErrC = 0 , DevT = 0		SIMULATION	DSP56000/1
(NC,NT) 4,4	$\| \delta C \|_2$	2.499300E-03	2.459738E-03
	$\| \delta C \|_{max}$	7.985590E-03	7.935523E-03
(NC,NT) 5,5	$\| \delta C \|_2$	1.215720E-03	1.212131E-03
	$\| \delta C \|_{max}$	3.692390E-03	3.674507E-03
(NC,NT) 6,6	$\| \delta C \|_2$	4.946470E-04	5.084432E-04
	$\| \delta C \|_{max}$	1.793150E-03	1.835823E-03
(NC,NT) 7,7	$\| \delta C \|_2$	2.716800E-04	2.779828E-04
	$\| \delta C \|_{max}$	1.077650E-03	1.084328E-03
(NC,NT) 8,8	$\| \delta C \|_2$	1.600700E-04	1.870916E-04
	$\| \delta C \|_{max}$	6.846190E-04	1.281480E-04

CHAPITRE IV

INTERPRÉTATION DES RÉSULTATS DES APPLICATIONS

Nous venons de faire une étude de quelques possibilités d'applications du processeur numérique de signaux (DSP56000/1) pour la résolution des problèmes liés à la reconstitution des signaux de mesure.

Ce chapitre s'attarde à analyser les résultats et les avantages de l'utilisation du processeur numérique de signaux ainsi que leurs apports, pour les trois applications décrites précédemment (§ III-1; § III-2 et § III-3).

En effet, l'analyse des erreurs, dans une conversion analogique numérique de plusieurs signaux multiplexés, a confirmée la pertinence d'une étude des possibilités de leur correction en temps réel. Les filtres développés à l'aide des polynômes de Lagrange sont utiles pour la correction de ces erreurs. Les résultats de correction (fig.3.4-e à fig3.4-f et fig.3.5-e à fig.3.5-b) montrent la pertinence du processeur numérique face aux exigences de traitement en temps réel.

L'étude faite sur la résolution du problème de déconvolution, montre qu'il est possible de reconstituer les signaux de mesure, affectés par le bruit, si on introduit les informations a priori sur le bruit dans la solution finale afin de réduire son effet, en utilisant la technique de régularisation de Tikhonov. Les résultats pratiques (fig.3.23-a, fig3.23-b, fig.3.24-a, fig.3.24-b, fig.3.25-a et fig.3.25-b) présentent l'intéret en

spectroscopie, de permettre l'amélioration de la résolution des signaux spectrométriques.

Enfin, pour la dernière application, les résultats pratiques et de simulations de l'étalonnage statique du système de mesure (tableaux 3.3.1-I à 3.3.1-IV et 3.3.2-I à 3.3.2-IV) montrent l'efficacité des fonctions d'interpolation "spline" pour la modélisation du système. L'erreur entre les concentrations références et celle calculées par le spline, est acceptable.

Quant à l'apport du processeur numérique de signaux pour ces applications, on peut le décrire de la façon suivante :

L'importance de l'utilisation du processeur numérique de signal, pour la première application, réside dans la correction en temps réel des erreurs de multiplexage, une tache qui n'aurait pu être facile si on a utilisé un microprocesseur à usage général. En effet, une correction d'un échantillon avec le DSP56000 (pour le type du filtre utilisé) ne demande que 0.6 μs, donc on peut aller jusqu'a 1 MHz comme fréquence d'échantillonnage des signaux. Alors que si on a utilisé, par exemple, le microcontrôleur MC68HC11, la correction d'un échantillon demande 4.6 ms (toujours pour le même filtre), et la correction en temps réel ne sera possible qu'avec une fréquence d'échantillonnage inférieur à 200 Hz. Ce qui explique l'efficacité de l'utilisation du processeur numérique pour ce type d'application.

La deuxième application présente une certaine particularité, qui est la grande quantité de calcul de FFT demandée pour reconstituter le signal de mesure. En effet, dans l'algorithme de reconstitution, il faut faire deux fois la transformée de Fourier directe, une fois la convolution et une fois la transformée de Fourier inverse, auxquelles s'ajoutent les autres opérations arithmétiques classiques, et cela quand on calcule la valeur de α séparément et on introduit sa valeur dans le programme

principal. Il s'en suit, l'importance d'utilisation d'un circuit qui possède une grande capacité de calcul avec une meilleur exactitude de traitement des données, deux critères un peu difficile de les trouver avec les microprocesseurs à usage général. Grace à ces performances et sa rapidité de traitement de données, le processeur numérique de signal offre ces services pour ce genre de calcul. En effet, pour réaliser l'opération de FFT rapide, le DSP56000 demande moins que 1 ms pour faire 128 points complexes, alors qu'avec le microcontrôleur MC68HC11 ca prend plus que 20 ms. En plus on a 24 bits pour présenter les données de mesure avec le DSP56000, ce qui offre une bonne plage dynamique et une meilleur exactitude de présentation des données. Il est, donc, totalement indispensable pour la réalisation pratique de cette application.

La présence du processeur numérique, pour la troisième application, n'est pas totalement indispensable comme pour les deux autres applications. En effet, on peut se contenter d'un microcontrôleur ou microprocesseur, vu que notre but est de trouver une fonction d'approximation pour le modèle du système de mesure à partir des données expérimentales, on n'aura pas besoin donc de traitement en temps réel là dedans, ni des opérations complexes comme la FFT, ce sont uniquement des opérations classiques. L'avantage de l'utilisation du processeur dans cette application est l'exactitude de calcul (24 bits de données) et aussi la facilité de programmation et de traitement en parallèle. Cependant, le DSP56000 utilise une unité arithmétique à point fixe, ce qui oblige l'utilisation des procédures de programmation à points flottants qui vont augmenter le temps de calcul pour le traitement des données de mesure.

CONCLUSION

Les progrés enregistrés dans la technologie de fabrication des semiconducteurs ont facilité l'émergence des processeurs numériques de signaux. Graçe à leur rapidité de calcul, utilisant des unités arithmétiques à virgule fixe ou à virgule flottante, répondant ainsi aux critères et besoins du traitement en temps réel, leur champ d'application devient de plus en plus vaste dans tous les domaines concernés par l'exploitation et le traitement de signaux.

En effet, une étude de quelques applications du processeur numérique DSP56000 de Motorola, dans le domaine de reconstitution des signaux de mesure, a été élaborée.

L'analyse des erreurs dans une conversion A / N de la tension électrique, a confirmé la pertinence des possibilités de leur correction en temps réel.

La méthode de régularisation de Tikhonov offre une solution trés efficace pour la résolution du problème de déconvolution. Cette application présente l'intéret en spectroscopie de permettre la séparation des raies du spectre mesuré. Le processeur numérique de signal est totalement indispensable pour la réalisation pratique de cette application vu la grande quantité de calcul de FFT demandée.

Enfin, l'étalonnage d'un système de mesure électronique en utilisant les fonctions d'interpolations spline peut donner de bons résultats (de point de vue exactitude) si on utilise un processeur.

Les applications élaborées dans ce travail montrent la pertinence du processeur numérique de signal face aux exigences de traitement en temps réel et de l'exactitude du calcul. Ces applications ouvrent la voie à d'autre applications, car les processeurs sont capables de fournir de multiples fonctions, souvent demandées pour une seule application. Grace à leur performance, ils peuvent s'appliquer aux domaines où la dynamique peut temporairement devenir trés important et à ceux où une trés grande exactitude des coéfficients est requise. Ils laissent donc entrevoir un élargissement considérable du champ d'application dans le domaine d'instrumentation.

En conclusion, les applications élaborées, dans ce mémoire, représentent une large classe d'applications possibles dans le domaine de reconstitution de mesurande. Cela confirme l'applicabilité du DSP, et par extrapolation logique de processeurs spécialisés (dédiés) dans les systèmes de mesure. Les résultats de ce travail constituent une contribution à l'intégration des systèmes de mesure. L'intégration fonctionnelle et technologique représente une tendance actuelle dans le développement du domaine de systèmes de mesure.

BIBLIOGRAPHIE

ASCH, G. & al. " Les capteurs en instrumentation industrielle " Edition Dunod, Paris-1987.

BARWICZ, A. " A System Approach to Electric Measurement Helps in understanding Measurement Methods " IEEE trans. on Inst. and Meas. Vol. 1M-34 - Nº 4 Part 1- Decembre 1985.

BARWICZ, A & DION, J.L " Exigences d'une structure des mesures électroniques dans l'analyse ultrasonore des solutions " Actes du congrés Canadien en génie électrique et informatique - Vancouver 3-4 Novembre 1988.

BARWICZ, A et al " Calibration of an Electronic Measuring System for Ultrasonic Analysis of Solutions" Proc. IEEE Inst. and Meas. Technology Conference IMTC'90 -San Jose 13-15 Fevrier 1990. pp.100-102.

BELLEMARE,D. " Analyse et correction des erreurs dans la conversion analogique à numérique de la tension électrique " Mémoire de maîtrise en Electronique Industrielle. U.Q.T.R 1989.

BIRAUD, Y.G " Les méthodes de déconvolution et leurs limitations fondamentales " Revue de physique appliquée - Tome 11 ; Mars 1976 pp.203-214

BLUM, E.K " Numerical Analysis and Computation- Theory and Practice" Addison Wesley - 1972

DE COULON,F. " Théorie et traitement des signaux " Traité d'électricité Volume VI, Presses Polytechniques Romandes- 1984

EL-SHARKAWY, M. " Real Time Digital Signal Processing Applications with Motorola's DSP56000 Family ". Prentice Hall, Englewood Cliffs, New Jersey 1990.

HAMMING, R.W. " Numerical Methods for Scientists and Engineers ", second edition, Mc Graw-Hill, Inc. 1973

HUNT, B.R" Image Restoration in Digital Image Processing Techniques Academic Press - New York 1984

JANSSON, P.A " Deconvolution with Application in Spectroscopy " Academic Press, Inc 1984.

KUNT, M. " Traitement numérique des signaux " Traité d'électricité Vol. XX, édition GEORGI Suisse- 1980.

LIM, J. & OPPEINHEIM, A.V. "Advanced Topics in Signal Processing", Prentice Hall Inc. Englewoods Cliffs, New Jersey -1988.

MAX, J et al " Méthodes et techniques de traitement de signal et applications aux mesures physiques " Tome I- Masson Paris 1985.

MIEKINA,A. & MORAWSKI,R.Z"Noise Reduction in Spectrophotometric Data Processing using Tikhonov's Regularisation" Proc. 1st Int. IMEKO-TC4 Symp. on Noise in Electrical Measurement- Coms, 19-21 Juin 1986 pp. 217-222

MORAWSKI,R.Z "Unified Approach to Measurement Signal Reconstruction " Proc. IEEE Inst. and Meas. Technology Conference IMTC'89- Washington.DC 25-27 Avril 1989 pp.446-449.

MORAWSKI, R.Z " Basic Problems of Measurement Signal Reconstruction " Proc. Australasian Instrumentation and Measurement Conference AIM'89 (Adelaide, November 14-16, 1989).

PARATTE, P.A & ROBERT, P. " Systèmes de mesure " Presses Polytechniques Romandes - 1986.

SABATIER, P.C. " Application de la théorie de l'inversion " Revue du Cethedec - Ondes et signal Nº 76 1983. pp.1-18.

SCHAFER, R.W & RABINER, L " A Digital Signal Processing Approach to Interpolation " Proceedings of the IEEE - Vol. 61 Nº 6 1973

SCHUMAKER, L.L " Spline functions - Basic theory " New York, J.Wiely et Sons, Inc 1981.

SRIDHARAN, S. & DICKMAN, G. " Block Floating-point Implementation of Digital Filters using the DSP56000 " Microprocessors and Microsystems , Vol.12 Nº 6, July/August 1988, pp.299-308.

TIKHONOV,A.N & Al. " Méthodes de résolution de problèmes mal posés " , Moscou: Mir, 1976 (traduction du russe).

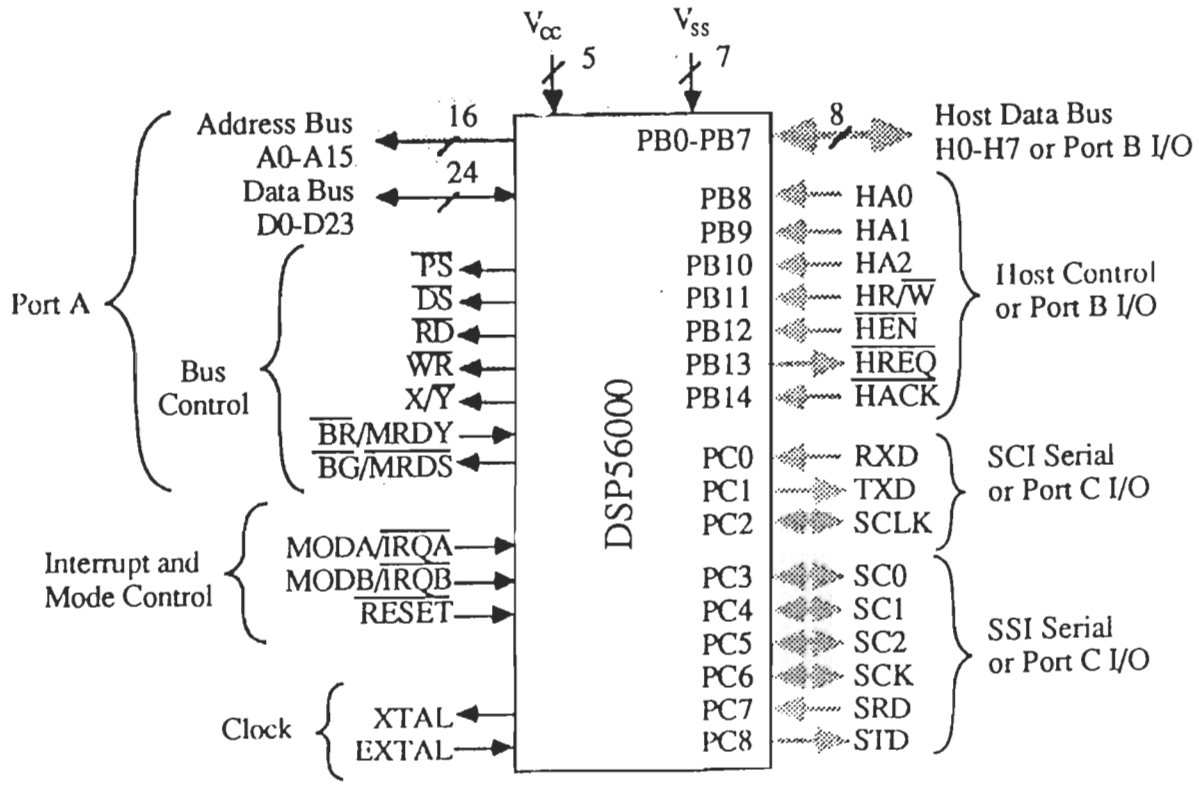
User's manual of DSP56000/56001 Digital Signal Processor- Motorola 1987.

ANNEXES

ANNEXE A-1

PROCESSEUR NUMÉRIQUE DE SIGNAUX

(DSP56000/1)



I- INTRODUCTION

La rapidité et la haute performance de traitement dans un environnement d'entrées/sorties en temps réel est la raison de naître un processeur de signal. Pour cela les processeurs numériques de signaux (ang. Digital Signal Processor) sont optimisés pour exécuter des algorithmes pour le traitement numérique de signaux avec le minimum d'opérations le plus possible tout en gardant une haute exactitude.

Malheureusement, pour beaucoup de processeurs le jeu d'instructions est parfois difficile d'emploi, le nombre de registres de données est insuffisant et les possibilités d'interruption sont limitées.

Premier élément d'une famille HCMOS, le DSP56000/1 de Motorola apporte une solution nouvelle à ces problèmes.

II-ORIGINE DE L'ARCHITECTURE DU PROCESSEUR DE SIGNAL

Depuis le début de notre décénie, les progrés enregistrés dans la technologie de fabrication des semiconducteurs et plus particulièrement la possibilité d'intégrer un nombre toujours plus important de transistors sur une même puce de silicium ont facilité l'émergence des microprocesseurs et des microcontrôleurs basés sur l'architecture de Von Neumann.

Malgrés l'amélioration de leurs performances, les microprocesseurs n'étaient pas tout à fait adaptés aux applications où les exigences de rapidité et d'exactitude étaient trop importante telque le traitement numérique en temps réel. Il a fallu donc élaborer un nouveau style d'architecture, et développer de nouveaux composants avec des vitesses de fonctionnement toujours plus rapide. De même que pour les microprocesseurs, l'integration croissante , en passant d'une technologie MOS à du NMOS puis du CMOS, et du circuit LSI au VLSI, a fait naître des générations de processeurs numériques de signaux.

III-CARACTÉRISTIQUES DU DSP56000/1

	cycle instruction	97,5 ns	
	taille des mots programme	24 bits	
	taille des mots données	24 bits	
	bus de données internes	4 x 24 bits	
	bus d'adresses internes	3 x 16 bits	
MEMOIRE INTERNE	RAM données	2 x 256 mots	
	ROM données	2 x 256 mots	pré-programmées dans DSP 56001
	mémoire programme	ROM 2048 mots pour DSP 56000	
		RAM 512 mots pour DSP 56001	avec bootstrap ROM de 32 mots
	taille du multiplieur	24 x 24 bits	
	registre d'entrée du multiplieur	4	
	taille de l'ALU	56 bits	
	accumulateurs	2 x 56 bits	
	temps de mult-accum. (MAC)	97,5 ns	
	registres d'adresses	24	
	hauteur de la pile	15 x 32 bits	
	sources d'interruptions	18	32 vecteurs ; 4 niveaux programmables
	taille du compteur de boucle	16 bits	7 niveaux d'imbrication
	taille du registre adresse de boucle	16 bits	de boucle DO
ESPACE MEMOIRE EXTERNE	bus données externes	1 x 24 bits	l'adressage externe ne nécessite pas d'instructions spécifiques
	bus d'adresses externes	1 x 16 bits	
	espace mémoire données	2 x 64 Kmots	– le temps d'accès sur le bus externe est programmable sur 4 zones indépendantes
	espace mémoire programme	64 Kmots	
ENTREE-SORTIE & PERIPHERIQUES	périphériques séries	SSI : synchrone	– toutes les E/S et tous les périphériques sont dans l'espace mémoire de données ainsi que leurs registres de contrôles,
		SCI : asynchrone	
	périphériques parallèles	HOST/DMA ; 8 bits	
	drapeaux E/S	24	– un jeu d'instructions spécifiques teste les E/S

IV- ARCHITECTURE INTERNE DU DSP56000/1

Le DSP56000/1 est basé sur une architecture Harvard hautement parallèle et non pipeline. Une telle architecture désignée pour maximiser la capacité intensive en données pour les applications du DSP.

La configuration interne du DSP56000/1 est dûale de nature, c'est qu'il y a deux espaces mémoires de données extensible, deux unités arithmétiques de génération d'adresses, une unité arithmétique et logique ayant deux accumulateurs et deux circuits décaleur-limiteur.

Le schéma bloc interne du DSP56000 est présenté à la figure 1. Toutefois il est constitué des circuits suivants :

- Quatre bus de données,
- Trois bus d'adresses,
- Unité arithmétique de génération d'adresses (AGU),
- Unité arithmétique et logique de données (ALU),
- Mémoire de données X,
- Mémoire de données Y,
- Contrôleur programme,
- Mémoire de programme,
- Entrées / Sorties
    - Extension de mémoire (Port A)
    - Entrées/Sorties (Port B et Port C),
    - Interface hôte,
    - Interface de communication série (SCI),
    - Interface série synchrone (SSI),

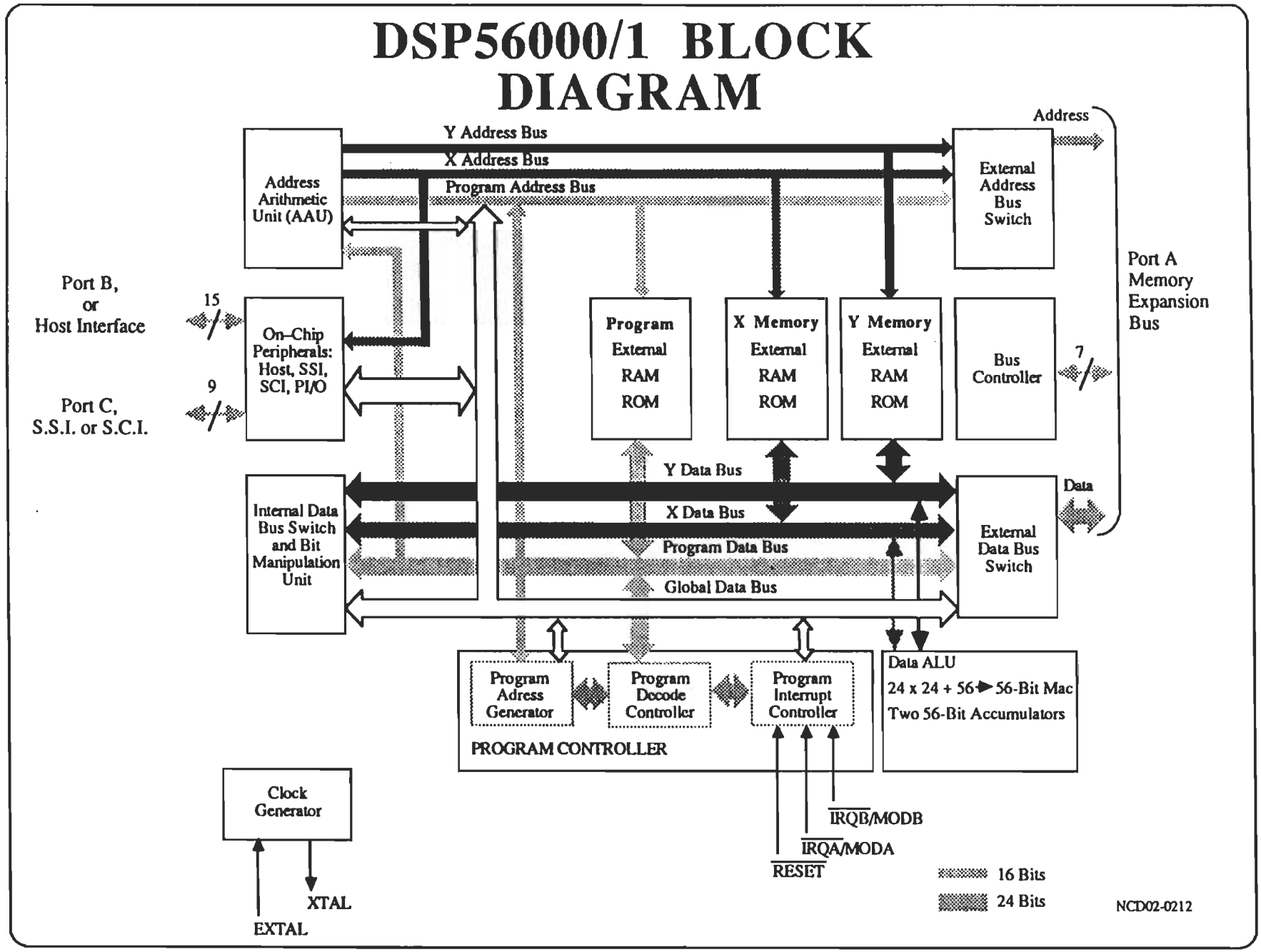


Fig.1 Architecture interne du DSP56000

IV-1 BUS DE DONNÉES

Les mouvements de données se font sur quatre bus bidirectionnels de 24 bits chacuns :

- Bus de données X (XDB),
- Bus de données Y (YDB),
- Bus de données du programme (PDB),
- Bus de données global (GDB),

Les bus XDB et YDB peuvent être traités dans certains instructions comme un seul bus de 48 bits en faisant la concaténation de ces deux bus.

Le transfert de données entre l'ALU et la mémoire de données X (respectivement Y) se produit à travers le bus XDB (respectivement YDB) Toutes les autres transferts de données, tel que le transfert des entrées-sorties avec les périphéries, se produisent à travers le bus GDB. La préextraction des mots d'instructions se produisse en parallèle sur le bus PDB. Cette structure de bus supporte le mouvement des données de registre à registre, registre à mémoire et mémoire au registre, et peut faire le transfert de trois mots (24 bits) ensemble dans un même cycle d'instruction. Le transfert entre les bus est accompli dans le commutateur de bus interne.

a- Commutateur de bus de données interne

Il est similaire à une matrice de commutation, et peut connecter deux bus internes sans introduire des retards de pipeline.

b- Unité de manipulation de bit

Elle est physiquement localisée dans le commutateur de bus de données interne, puisqu'il a l'accés à chaque espace mémoire. Cette unité exécute l'opération de manipulation de bit sur les opérandes mémoires de XDB, YDB et GDB.

IV-2 BUS D'ADRESSES

Le DSP56000 possède trois bus d'adresses, de 16 bits chacuns, répartis de la façon suivante :

- Un bus unidirectionnel pour la mémoire de données X (XAB),
- Un bus unidirectionnel pour la mémoire de données Y (YAB),
- Un bus bidirectionnel pour la mémoire du programme (PAB).

Les espaces mémoires externes sont adressés via un simple bus d'adresse unidirectionnel de 16 bits à travers le commutateur de bus d'adresse externe qui peut choisir XAB ou YAB ou PAB. Une seule mémoire externe est disponible par cycle.

IV-3 MÉMOIRE DE DONNÉES X

La RAM de données X occupe les 256 premiers octets de l'espace mémoire X. La ROM occupe les 256-512 octets. Les périphéries d'entrées-sorties occupent les 64 derniers octets de la mémoire X. Elle peut être extensible à l'extérieur, on aura donc 65535 positions mémoires accesssible.

IV-4 MÉMOIRE DE DONNÉES Y

La RAM de données Y occupe les 256 premiers octets de l'espace mémoire Y. La ROM occupe les 256-512 octets. Les périphéries d'entrées-sorties occupent les 64 derniers octets de la mémoire Y. Elle peut être extensible à l'extérieur, on aura donc 65535 positions mémoires accesssible.

IV-5 MÉMOIRE PROGRAMME

La mémoire programme implantée sur le circuit comprenne 3.75

Kmots de ROM pour le DSP56000 ou 512 mots de RAM pour le DSP56001. Les adresses sont reçues du programme de contrôle logique(le compteur programme), les adresses des vecteurs d'interruptions sont localisées dans les 64 derniers octets de l'espace mémoire programme.

IV-6 UNITÉ ARITHMÉTIQUE ET LOGIQUE DE DONNÉES

Elle consiste (figure 2) de:

- Quatre registres d'entrées de 24 bits,
- Deux accumulateurs de 48 bits,
- Deux accumulateurs d'extension de 8 bits,
- Un accumulateur de décalage,
- Deux registres décaleur-limiteur,
- Une unité de multiplication-accumulation (MAC).

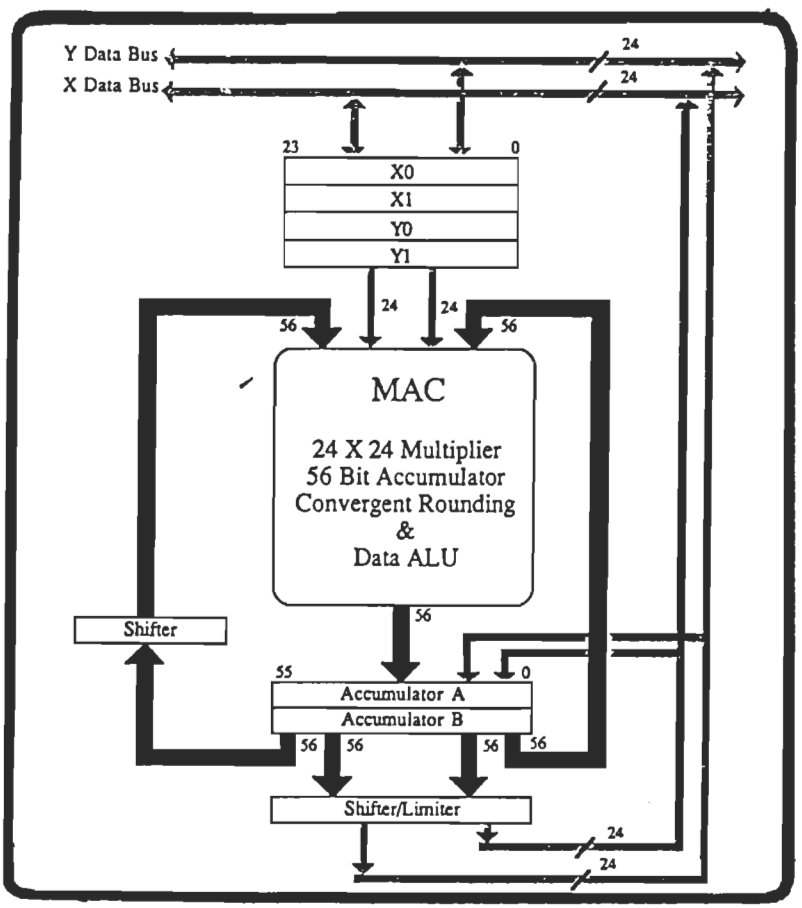


Fig.2 : Unité arithmétique et logique du DSP56000

IV-6-1 Registres d'entrées des données ALU

X1, X0, Y1, Y0 sont les quatres registres d'entrées de données. Ils peuvent être traités comme 4 registres indépendants de 24 bits ou comme deux registres dits X et Y suite à la concaténation de X1:X0 et Y1:Y0.

Ces registres servent comme registres tampons entre XDB ou YDB et l'unité de multiplication-accumulation (MAC). Ils sont utilisés comme source de données ALU permettant la nouvelle opérande d'être chargé à l'instruction suivante pendant que les contenus des registres sont utilisés par l'instruction en cours.

IV-6-2 Registres d'accumulateurs

Les six registres d'accumulateurs forment deux accumulateurs de 56 bits A et B. Chaque accumulateur comprend trois registres d'accumulation concatenés ensemble (A2 : A1 : A0 et B2 : B1 : B0 respectivement) de la façon suivante :

A	=	A2	:	A1	:	A0
56 bits	=	8 bits	:	24 bits	:	24 bits
B	=	B2	:	B1	:	B0
56 bits	=	8 bits	:	24 bits	:	24 bits

Les 24 bits du produit le plus significatif (MSP) sont stockés dans A1 ou B1 et les 24 bits du produit le moins significatif (LSP) sont stockés dans A0 ou B0. Les 8 bits d'extension, qui offrent une protection contre le dépassement, sont stockés dans A2 ou B2.

IV-6-3 Unité de multiplication-accumulation (MAC)

Cette unité, comme le nom l'indique, fait la multiplication de deux données de 24 bits, venant des registres d'entrées X ou Y puis accumule

le résultat au contenu de l'accumulateur A ou B, et le stocke au même accumulateur et cela en une seule cycle machine. Cette opération est trés utile dans le traitement numérique des signaux , surtout dans le cas du filtrage numérique et du transformée de Fourier.

Si la multiplication sans accumulation est spécifiée dans l'instruction, l'unité MAC initialise l'accumulateur et aprés il ajoute le résultat de multiplication. Les résultats arithmétiques obtenus de cette unité sont de 56 bits. L'arrondissement convergente est exécuté au moment de l'accumulation s'il est specifié dans l'instruction (e.g : l'instruction MAC et MACR), le bit de l'accumulateur qui sera arrondi est spécifié par le bit du mode " scaling" (changement d'echelle) du registre d'état.

IV-6-4 Accumulateur décaleur

C'est un décaleur parallèle asynchrone de 56 bits d'entrée et 56 bits de sortie, appliqué immédiatement avant les accumulateurs d'entrées de l'unité MAC. Les opérations qui font appel à cet accumulateur sont:

- Pas de décalge,
- Décalage à gauche d'un bit ASL, LSL, ROL,
- Décalage à droite d'un bit ASR, LSR , ROR,
- Forcer à zéro.

IV-7 UNITÉ DE GÉNÉRATION D'ADRESSES (AGU) (fig 3)

Tous les calculs des adresses éffectives, nécessaire pour adresser les opérandes de données dans la mémoire, sont exécutés dans l'unité arithmétique de génération d'adresses. Cette unité travaille en parallèle avec les autres circuits afin de minimiser les temps système de génération d'adresses, et elle applique trois types d'arithmétiques :

- arithmétique linéaire,
- arithmétique modulo,
- arithmétique à retenu inverse.

L'AGU contienne huit registres d'adresses (R0,R1,R2,...,R7), huit registres d'offset (N0,N1,N2,...,N7) et huit registres de modification (M0,M1,M2,...,M7). Les 24 registres Rn, Nn et Mn sont traités comme des registres triplets(e.g : uniquement M2 et N2 peuvent être utilisés pour mettre à jour R2). Les huits triplets sont :

R0:M0:N0 , R1:M1:N1 , R2:M2:N2 , R3:M3:N3
R4:M4:N4 , R5:M5:N5 , R6:M6:N6 , R7: M7:N7

IV-7-1 Registres d'adresses (Rn)

Ils sont divisés en deux sous ensembles (R0,R1,R2.R3) et (R4,R5,R6,R7) qui contiennent toujours des adresses utilisées comme pointeur de mémoire. Le contenu de Rn peut pointer directement une donnée, il peut aussi être prémise à jour ou postmise à jour selon le mode d'adressage choisi. Chaque registre peut être lû ou écrit à travers le bus GDB, et il est préreglé à $FFFF durant l'initialisation du processeur. Au moment de la lecture par le GDB, les registres seront écrites dans les deux octets les moins significatifs du bus global, l'octet le plus significatif est mis à zéro. Au moment de l'écriture à partir du bus global, uniquement les deux octets les moins significatifs sont écrites, l'autre sera tronqué.

IV-7-2 Registres d'offset (Nn)

Ils sont divisés en deux sous ensembles (N0,N1,N2,N3) et (N4,N5,,N6,N7), et contiennent les valeurs d'offset utilisés pour la mise à jour des pointeurs d'adresses et de données. Par exemple le contenu d'un registre d'offset peut être utilisé pour le saut par étapes les éléments d'une table(e.g. 5 positions par étape pour la génération d'une sinusoide à partir de la table sinus), ou il peut spécifier l'offset dans une table, ou la base d'une table pour l'adressage indexé. Chaque registre Rn

possède son propre registre d'offset Nn associé à lui.

IV-7-3 Registres de modification (Mn)

Ils sont divisés en deux sous ensembles (M0,M1,M2,M3) et (M4,M5,M6,M7) et leurs contenus déterminent le type d'arithmétique utilisé . Ils peuvent contenir aussi des données. Sept modes d'adressage linéaire (Mn=$FFFF) indirectes sont permis. L'arithmétique avec propagation de la retenue est sélectionné par Mn=$0000, alors que Mn=k-1 sélectionne l'arithmétique modulo sur les registres d'adresses. Chaque registre d'adresse Rn possède son registre Mn propre à lui seul.

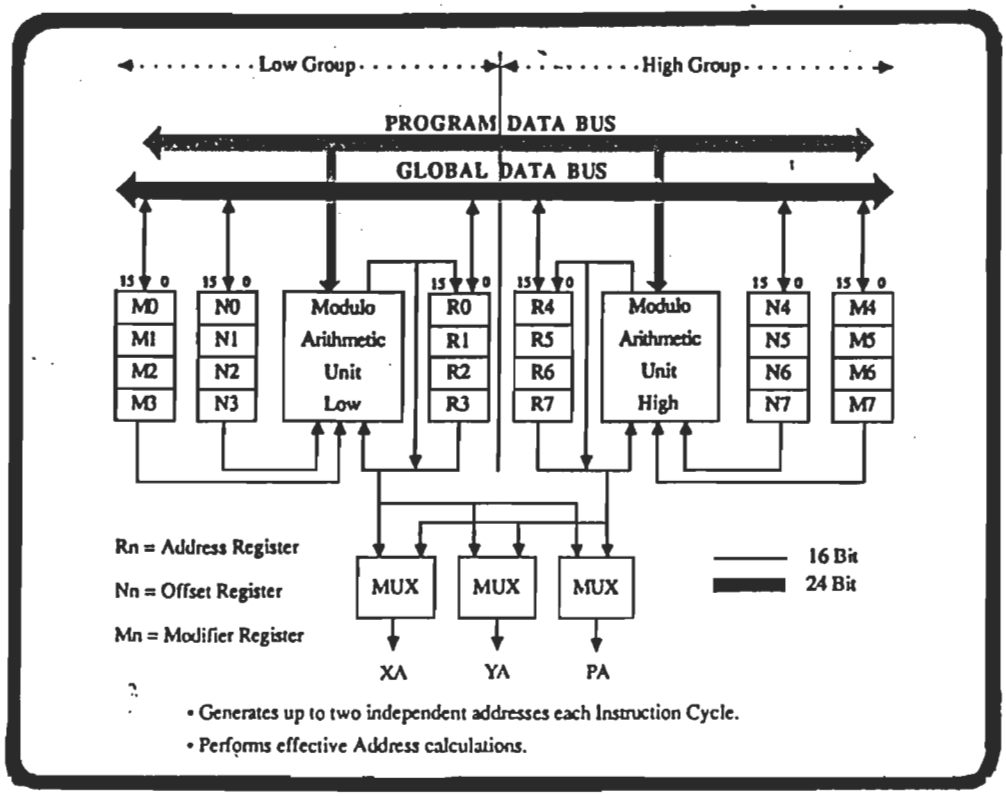


Fig.3 Unité artithmétique de génération d'adresse

IV-8 CONTROLEUR PROGRAMME (figure 4)

Le contrôleur programme exécute la génération d'adresse du programme (préextraction des instructions), décodage des instructions

et contrôle matérielle DO LOOP. Il comporte six registres et une pile système ("Stack System SS"). En plus des registres standards, tel que le compteur programme (PC) le registre d'état (SR) et le pointeur de pile(SP), le contrôleur programme possède les registres d'adresse de boucle (" Loop Adress LA ") et compteur de boucle (" Loop Counter LC") nécessaires pour l'instruction DO LOOP. La pile système SS possède 15 mots de 32 bits permet de mémoriser jusqu'a quinze interruptions longues, sept niveaux d'imbrication de l'instruction " DO ", quinze niveaux d'appel à des sous programmes. Chaque position dans SS est adressable comme registre de 16 bits SS haut (SSH) et SS bas (SSL) qui sont pointé par SP.

Architecture du contrôleur programme :

Le contrôleur programme comprend trois blocs matériels :

1-Le décodeur du contrôleur programme (PDC).

2-Le contrôleur d'interruption programme (PIC) : Il reçoit tous les demandes d'interruption, et génère les adresses des vecteurs d'interruption. Il y a quatre sources d'interruptions externes et 16 internes qui peuvent générer des interruptions. Ces interruptions sont organisées selon des niveaux de priorité, chacune est associée avec son niveau de priorité (IPL) qui peut être de 0 à 3. Les niveaux 0, 1 et 2 sont masquables et le niveau 3, qui est le plus haut, est non masquable (figure 5).

3-Le générateur d'adresse programme (PAG): Il contient le PC, SP, SS, le registre de mode d'exploitation(OMR), SR, LC et LA.

Le compteur programme (PC) contient l'adresse de la prochaine instruction et peut pointer des opérandes d'adresses et des opérandes de données.

En exécutant l'instruction " DO LOOP ", on charge LC par le nombre de fois que la boucle doit être répétée et on charge LA par l'adresse de la dernière instruction de la boucle.

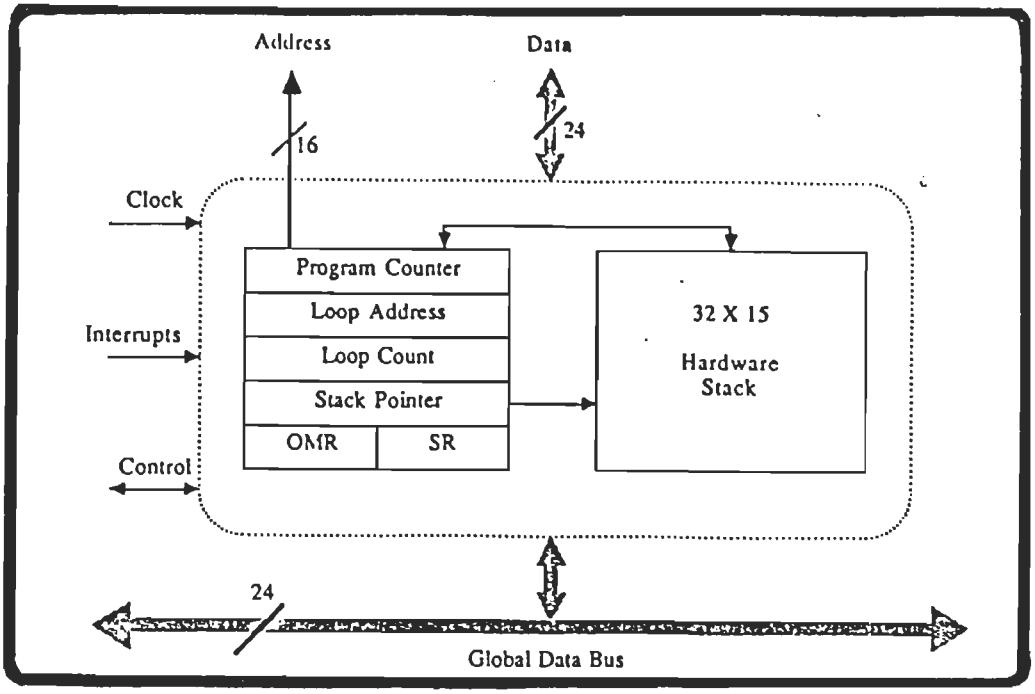


Fig.4 Le contrôleur de programme.

Interrupt Starting Address	IPL	Interrupt Source
P:S0000 or P:SE000	3	Hardware $\overline{\text{RESET}}$ (External)
P:S0002	3	Stack Error
P:S0004	3	Trace
P:S0006	3	SWI (Software Interrupt)
P:S0008	0–2	$\overline{\text{IRQA}}$ (External)
P:S000A	0–2	$\overline{\text{IRQB}}$ (External)
P:S000C	0–2	SSI Receive Data
P:S000E	0–2	SSI Receive Data with Exception Status
P:S0010	0–2	SSI Transmit Data
P:S0012	0–2	SSI Transmit Data with Exception Status
P:S0014	0–2	SCI Receive Data
P:S0016	0–2	SCI Receive Data with Exception Status
P:S0018	0–2	SCI Transmit Data
P:S001A	0–2	SCI Idle Line
P:S001C	0–2	SCI Timer
P:S001E	3	NMI — Reserved for Hardware Development (External)
P:S0020	0–2	Host Receive Data
P:S0022	0–2	Host Transmit Data
P:S0024	0–2	Host Command (Default)
P:S0026	0–2	Available for Host Command
P:S0028	0–2	Available for Host Command
P:S002A	0–2	Available for Host Command
P:S002C	0–2	Available for Host Command
P:S002E	0–2	Available for Host Command
P:S0030	0–2	Available for Host Command
P:S0032	0–2	Available for Host Command
P:S0034	0–2	Available for Host Command
P:S0036	0–2	Available for Host Command
P:S0038	0–2	Available for Host Command
P:S003A	0–2	Available for Host Command
P:S003C	0–2	Available for Host Command
P:S003E	0–2	Illegal Instruction

Fig.5 Vecteurs d'interruption et niveau de priorité

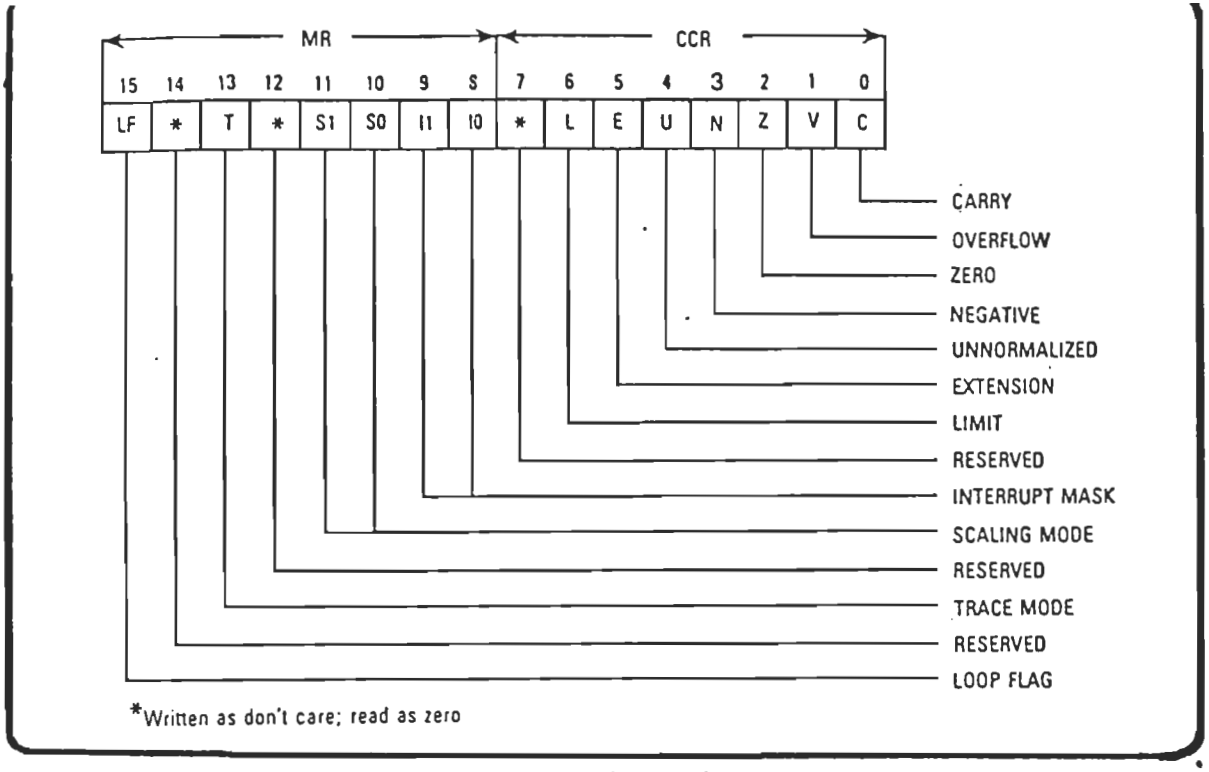


Fig.6 Le registre d'état du DSP56000/1

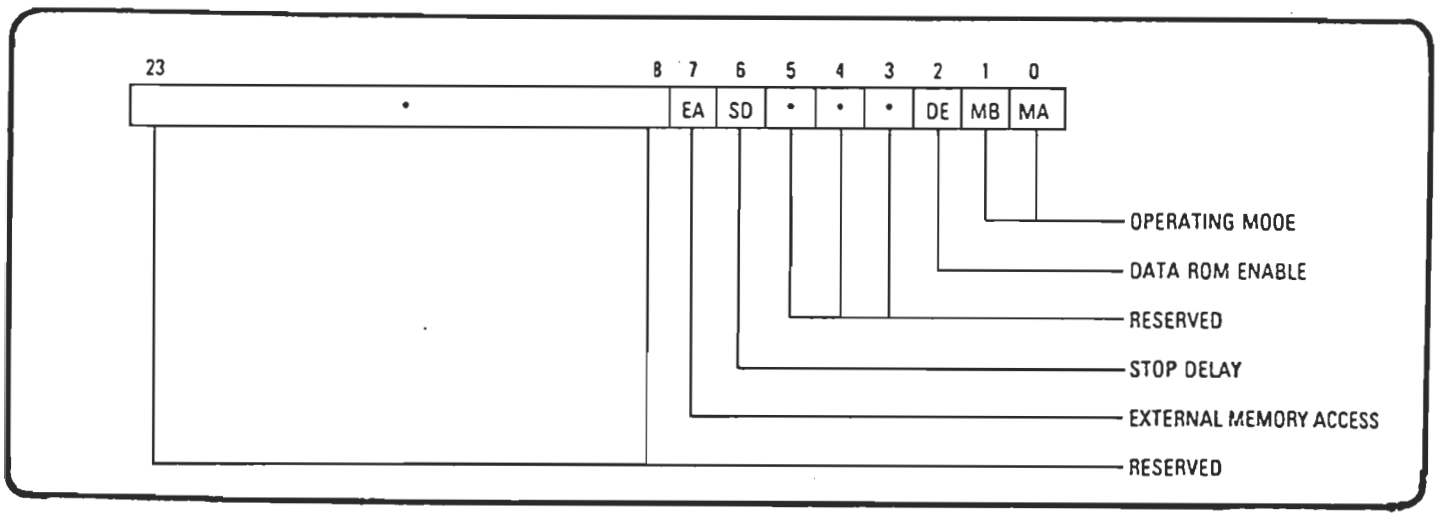


Fig.7 Le registre de mode d'exploitation du DSP56000/1

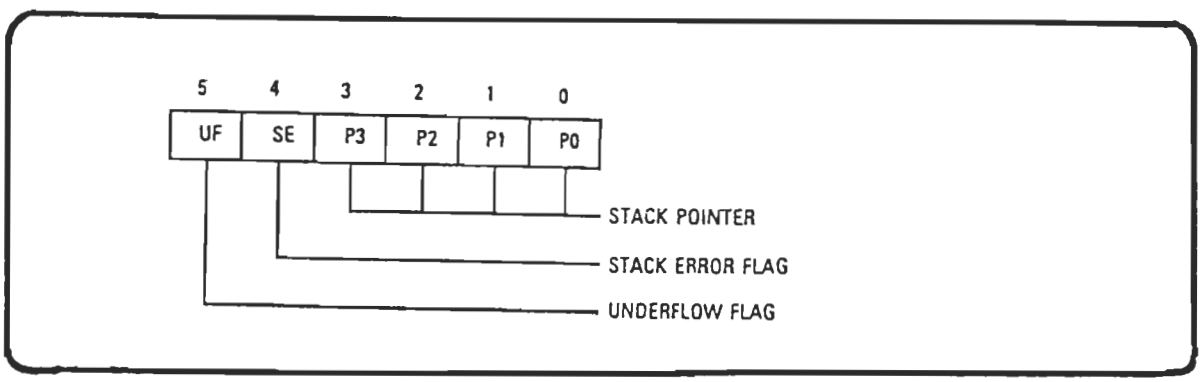


Fig.8 Le pointeur de pile du DSP56000/1

Ces deux registres sont empilés, dans SSH pour LA et SSL pour LC, par l'instruction " DO " et dépilés par l'instruction " ENDDO " ou bien à la fin du boucle.

Le registre d'état (figure 6) comprend un registre code condition (occupe les 8 bits les moins significatifs) et un registre de mode MR (occupe les 8 bits les plus significatifs). Le MR est un registre de contrôle définissant l'état courant du processeur.

Le registre de mode d'exploitation (OMR) (figure 7) est un registre 24 bits (uniquement 5 qui sont définis) qui détermine le mode d'opération et d'exploitation du processeur.

Le registre pointeur de pile SP (figure 8) est un registre de 6 bits indique la position top de SS et l'état de SS (dépassement négative, dépassement positive, vide, plein).

IV-9 LES ENTRÉES/ SORTIES (figure 9)

Les 24 broches indépendantes des bus d'adresses et données sont réparties en deux ports B (15 bits) et C (9 bits), qui peuvent être programmés en entrée-sortie à usage général, ou en trois périphériques. Tous ces périphériques fonctionnent en simultanné (" full duplex ") et possèdent des buffers (tampons) doubles, pour séparer la réception de la transmission. Ils peuvent être contrôlés par interruption ou par scrutation.

Utilisée comme entrée-sortie générale, chaque broche procure un drapeau (flag), dont la direction est programmée dans un registre de direction. Programmée en sortie, elle peut être modifiée par une instruction de manipulation de bits. Les instructions de test sur bit et de saut de test sur bit peuvent être utilisées sur les flags d'entrées. Le port d'accés direct à la mémoire (DMA) de l'unité centrale est un port 8 bits autorisant un transfert à la vitesse de 10 Mégaoctets/seconde. Les mixages de transfert de données sur 8, 16 et 24 bits autorisent le "

mapping " du DSP56000 avec tous les autres microprocesseurs classiques.

L'unité centrale peut d'une part, envoyer des interruptions vectorisées en écrivant dans le registre de vecteurs d'adresse et d'autre part, provoquer le branchement à l'une des 32 adresses de programme d'exception.

Le DSP56000 peut interrompre l'unité centrale par l'intermédiaire de la broche " HREQ ". Un membre de la famille MC68000 peut activer " HACK " pour lire un vecteur d'interruption. Le mode DMA est compatible avec les contrôleurs DMA se trouvant sur le marché.

Les trois broches du port asynchrone permettant un dialogue (débit allant jusqu'a 320 Kbits/sec) avec tout ACIA, UART, interface RS-232C ou modem. Son mode série à 2.5 Mégabits/sec est compatible avec le mode registre à décalage des 8051 et 8096. Il contient un générateur de baud et un temporisateur d'interruption.

Le port d'interface série synchrone nécessite entre trois et six broches, selon son mode d'utilisation. Son protocole réseau lui permet de s'interfacer directement à des multiplexes de cofidecs ou de DSP (les échanges sur 8, 16 ou 24 bits sont possibles). Le nombre de mots par trame peut alors être choisi entre 1 et 32. La transmission et la réception peuvent fonctionner en synchrone ou asynchrone, les horloges pouvant être externes ou générées en interne en divisant la fréquence du quartz dans un diviseur prescaleur de 8 bits .

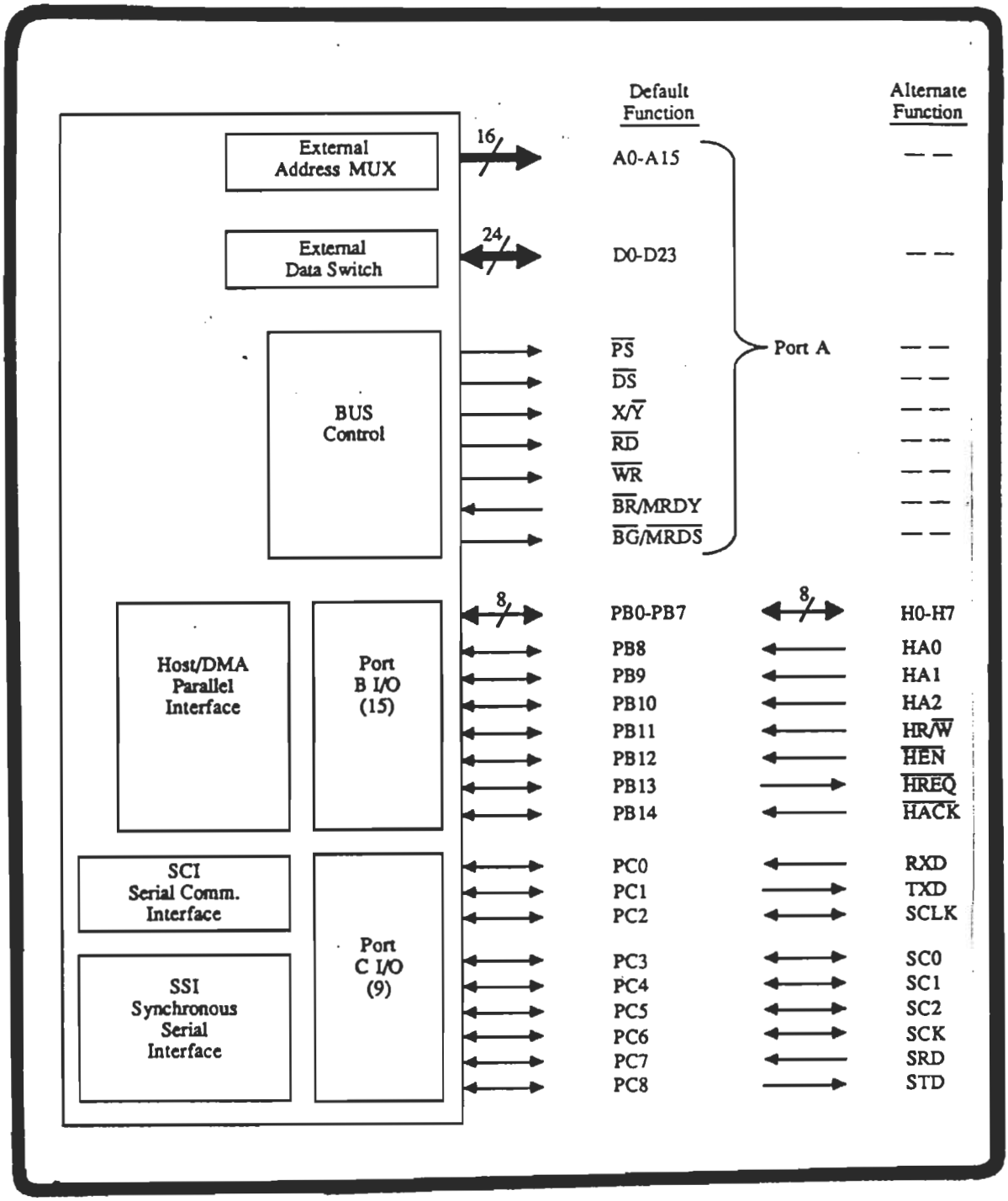


Fig.9 **Les entrées-sorties du DSP56000/1**

V- MODES D'EXPLOITATION DU DSP56000/1

Ils déterminent la répartition de la mémoire du processeur. Quand le DSP est à l'état Reset, les broches MODA/MODB sont lûes à ce que le DSP sort de cet état, à ce moment on a :

MODA/MODB ----------> IRQA/IRQB

Les modes d'exploitation peuvent être changées en modifiant l'état des bits du registre du mode d'exploitation (OMR).

V-1 Pour le DSP56000

On distingue quatre mode de fonctionnement selon l'état des broches MODA/MODB. :

- Mode Single chip (Mode 0 et Mode 1). (MODA =0 et MODB =0 pour le mode 0 et MODA=1 MODB =0 pour le mode 1. Cependant il est recommandé à l'utilisateur de ne pas utiliser le mode 1).
- Mode normalement étendu (Mode 2). (MODA = 0 et MODB =1)
- Mode de développement (Mode 3). (MODA =1 et MODB =1).

a- Mode Single Chip :Le contrôleur programme exécutera, aprés le reset, l'instruction se trouvant à l'adresse $0000 de la RAM programme interne (PRAM).

b- Mode normalement étendu : L'instruction à l'adresse externe P: $E000 est exécuté aprés le reset. Ce mode permettera le chargement de la RAM programme interne de manière trés rapide à partir d'une mémoire externe de 24 bits.

c- Mode de développement : C'est un mode dans lequel toute la mémoire est externe au circuit, et il peut être utilisé pour la réalisation des prototypes.

V-2 Pour le DSP56001

Le processeur de signal DSP56001 est une version dans laquelle les

2048 mots de ROM programme (DSP56000) ont été remplacés par 512 mots de RAM destinée au chargement d'un programme en utilisant un mode spécial.

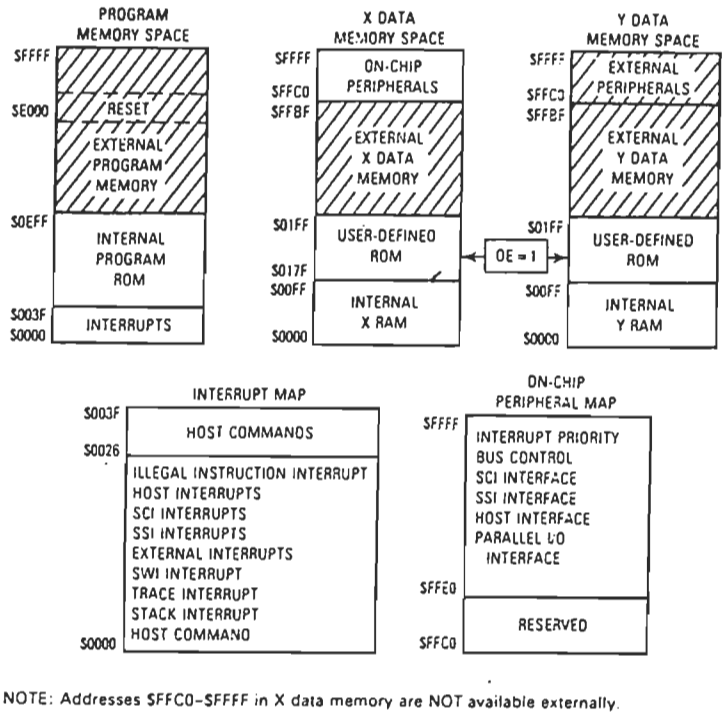
On distingue aussi quatre mode d'exploitation du DSP56001 :

- Mode Single Chip (Mode 0),
- Mode Bootstrop (Mode 1),
- Mode normalement étendu (Mode 2) (figure 10),
- Mode de développement (Mode 3).

N,B :Les modes 0, 2 et 3 sont identiques à ceux du DSP56000.

Mode Bootstrop : C'est le mode qui permet le chargement de la PRAM en exécutant le programme contenu dans la ROM du "bootstrop" masquée par le fabricant.

L'état du bit de poids fort (bit 23) de l'adresse P:$C000 détermine si le programme sera chargé octet par octet à partir du bus externe ou par l'intermédiaire du port Host/DMA.



NOTE: Addresses $FFC0–$FFFF in X data memory are NOT available externally.

Fig.10 Configuration de la mémoire pour le mode2

VI- MODES D'ADRESSAGE DU DSP56000/1

Le DSP56000/1 possède trois différents modes d'adressages :

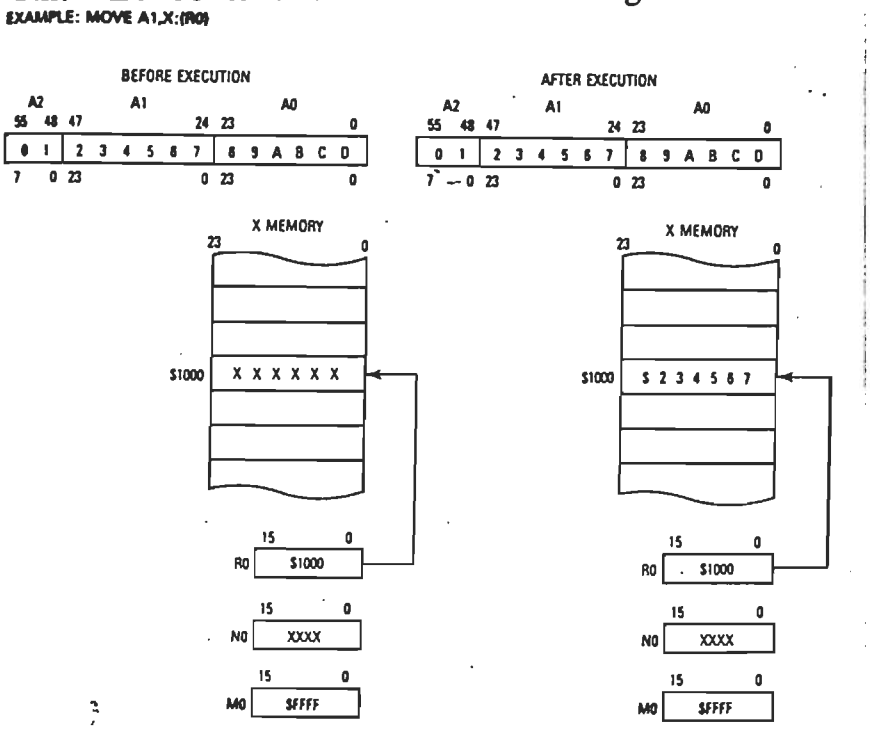
- Mode d'adressage direct des registres,
- Mode d'adressage indirect des registres d'adresses,
- Mode d'adressage spécial.

VI-1 Mode d'adressage indirect des registres d'adresses :

Quand un registre d'adresse est utilisé, pour pointer une position mémoire, le mode d'adressage est dit " Indirect ". Ce terme est utilisé car le contenu du registre n'est pas l'opérande lui même, mais plutôt l'adresse de l'opérande.

On distingue les différents modes suivants :

a-Pas de mise à jour : l'adresse de l'opérande est dans le registre d'adresse Rn. Le contenu de Rn reste inchangé en exécutant l'instruction.



b-Postincrément par 1 : le contenu de Rn est incrémenté par 1 aprés l'exécution de l'instruction .

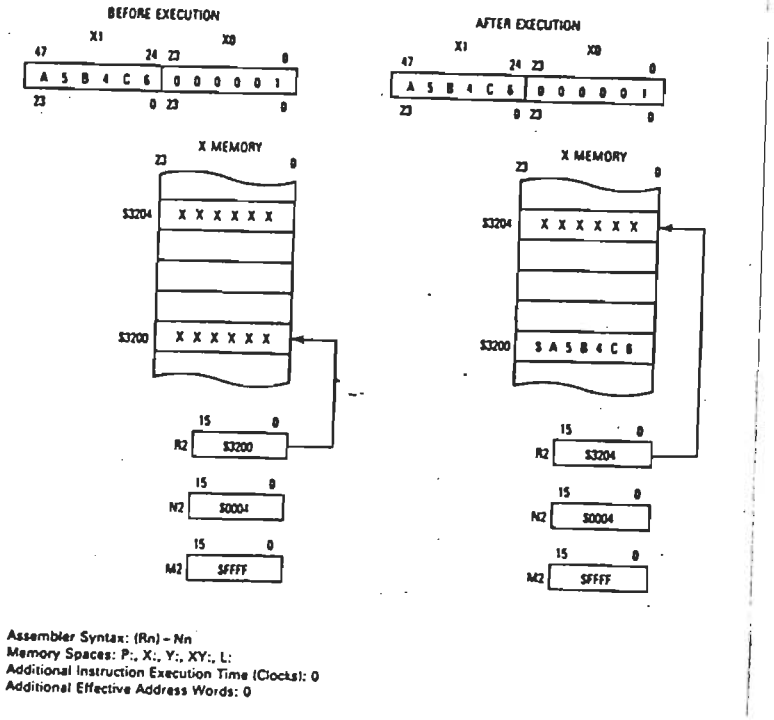
e.g : MOVE B0, Y:(R1)+

c-Decrément par 1 : le contenu de Rn est décrémenté par 1 aprés l'exécution de l'instruction.

e.g : MOVE Y0, Y:(R3)-

d-Post incrément par le contenu de Nn :aprés l'utilisation de l'adresse opérande, elle sera incrémentée par le contenu du registre Nn et stockée dans le même registre d'adresse. Le contenu de Nn reste inchangé.

e.g : MOVE X1, X:(R2)+N2



e-Décrément par le contenu de Nn : identique au précédent, sauf l'opération devienne la décrémentation .

e.g : MOVE X:(R4)-N4, A0

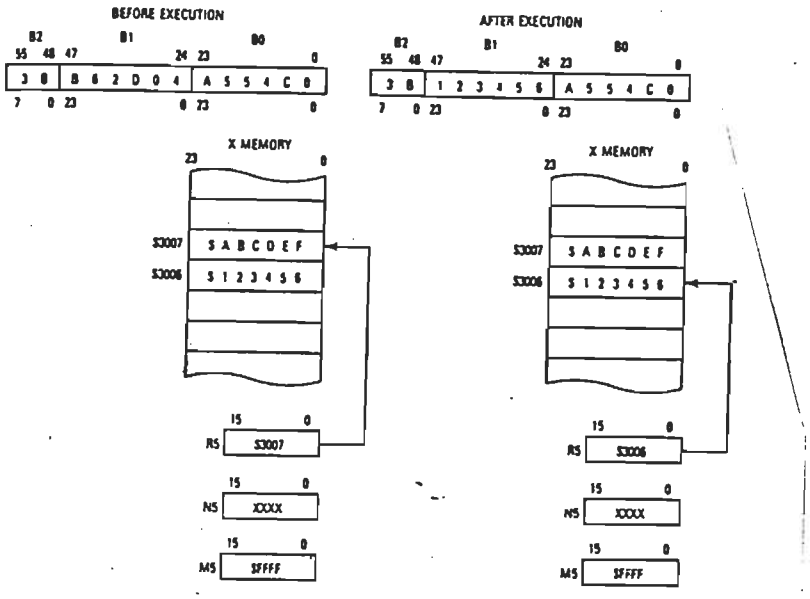
f-Indexation par le contenu de Nn : l'adresse de l'opérande est la somme du contenu du registre d'adresse Rn et le contenu du registre

d'offset Nn. Leurs contenus restent inchangés aprés l'exécution de l'instruction.

e.g : MOVE Y1, X:(R6 + N6)

g-Prédecrément de 1 : l'adresse de l'opérande est le contenu de Rn décrémenté de 1 avant que l'adresse de l'opérande soit utilisée. Le contenu de Rn est décrémenté et stocké dans le même registre d'adresse.

e.g : MOVE X:-(R5), B1



VI-2 Mode d'adressage direct des registres

Ce mode d'adressage effective spécifie que l'opérande source ou destination est l'une des registres de données, de contrôle ou d'adresses dans le modèle de programmation.

a-Mode direct des registres de contrôles ou de données: L'opérande peut être un, deux ou trois registres de données selon l'instruction. Ce mode d'adressage est utilisé aussi pour spécifier l'opérande du registre de contrôle pour quelques instructions spécilaes tel que ORI et ANDI.

b-Mode direct des registres d'adresses :L'opérande est l'une des 24 registres d'adresses (Rn, Nn ou Mn) spécifié par une adresse effective dans l'instruction.

VI-3 Mode d'adressage spécial :

a-Mode d'adressage immédiat des données

Ce mode d'adressage demande un seul mot de l'instruction contenant la donnée qui suit l'opérande immédiatement .

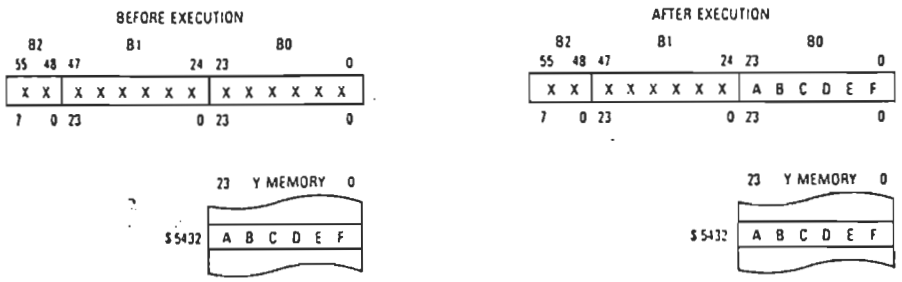
EXAMPLE A. IMMEDIATE INTO 24-BIT REGISTER
(MOVE #$123456,A0)

EXAMPLE B. POSITIVE IMMEDIATE INTO 56-BIT REGISTER
(MOVE #$123456,A)

EXAMPLE C. NEGATIVE IMMEDIATE INTO 56-BIT REGISTER
(MOVE #$801234,A)

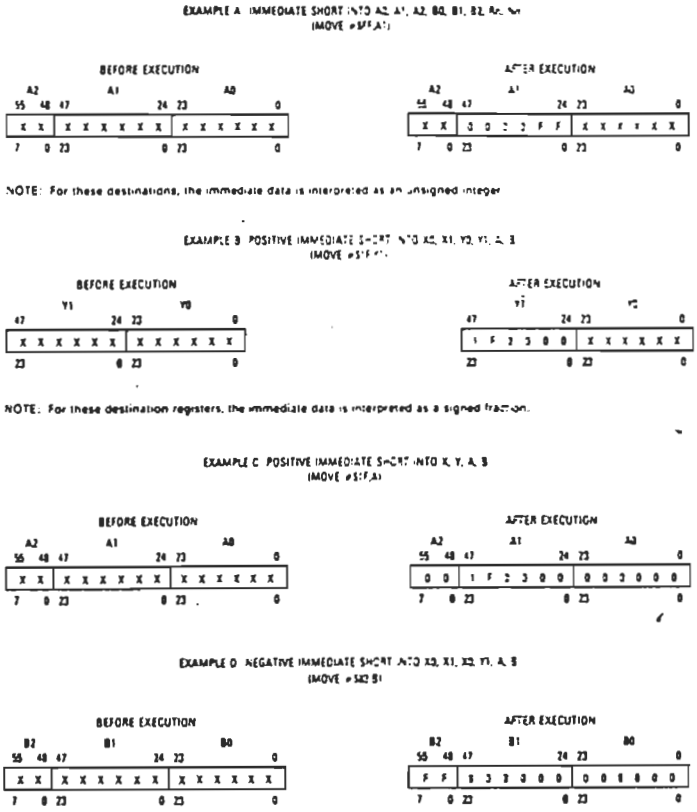
b-Mode d'adressage absolu

EXAMPLE: MOVE Y:$5432,B0

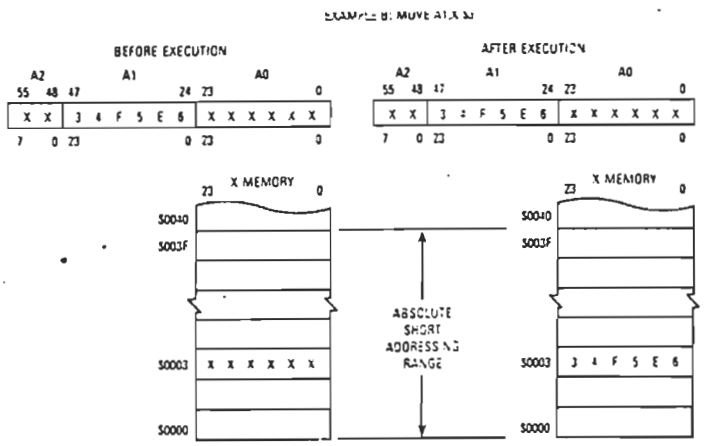


c-Mode d'adressage immédiat court

La donnée immédiate est interprété comme un entier non signé ou fraction signée dépendemment du registre de destination.



d-Mode d'adressage absolu court : L'adresse de l'opérande occupe 6 bits dans le mot d'instruction permettant ainsi une plage d'adressage de $0000 à $003F.

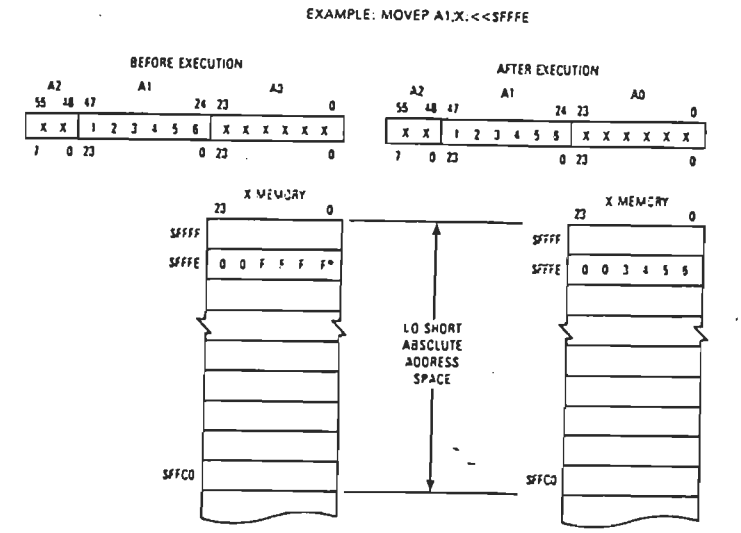


e-Mode d'adressage de saut court ("Short Jump"): L'opérande occupe 12 bits permettant ainsi uen plage d'adressage de $0000 à $0FFFF.

.e.g : JMP $123

f-Mode d'adressage d'entrée-sortie court :

Ce mode d'adressage est similaire à celui de l'absolue court, en ce qui concerne le format du mots (6 bits). Ce mode est utilisé avec le manipulation de bit et l'instruction MOVEP. L'adresse d'E/S court est de 1 à 16 bits pour adresser les espaces mémoires d'E/S de X et Y.



*Contents of Bus Control Register (X:SFFFE) After Reset

Assembler Syntax: pp
Operands Referenced: X, Y Memories
Additional Instruction Execution Time (Clocks): 0
Additional Effective Address Words: 0

g-Mode d'adresage implicite

Quelques instruction rend une référence implicite à PC, SS, LA, LC ou SR. Par exemple l'instruction JMP est implicitement référée à PC, alors que l'instruction REP est référé à LC.

VII- INSTRUCTIONS DU DSP56000/1

Le DSP56000 comporte toutes les instructions classiques et familières au traitement numérique des signaux (REP, MPY, ADD, SUB,...). Un certain nombre des 62 instructions sont cependant nouvelles et changent énormément les habitudes et la philosophie d'implantation des algorithmes.

On peut réparti les instructions du DSP en six groupes :(tableau I)

- instructions arithmétiques,
- instructions logiques,
- instructions de bit de manipulation,
- instructions de boucle,
- instructions de déplacement,
- instructions du programme contrôle.

EXEMPLE DE PROGRAMMATION

On va prendre l'exemple de multiplication de deux nombres complexes K1 et K2 avec :

$$K_1 = a_r + i\,a_i \quad \text{et} \quad K_2 = b_r + i\,b_i$$

le résultat de multiplication est :

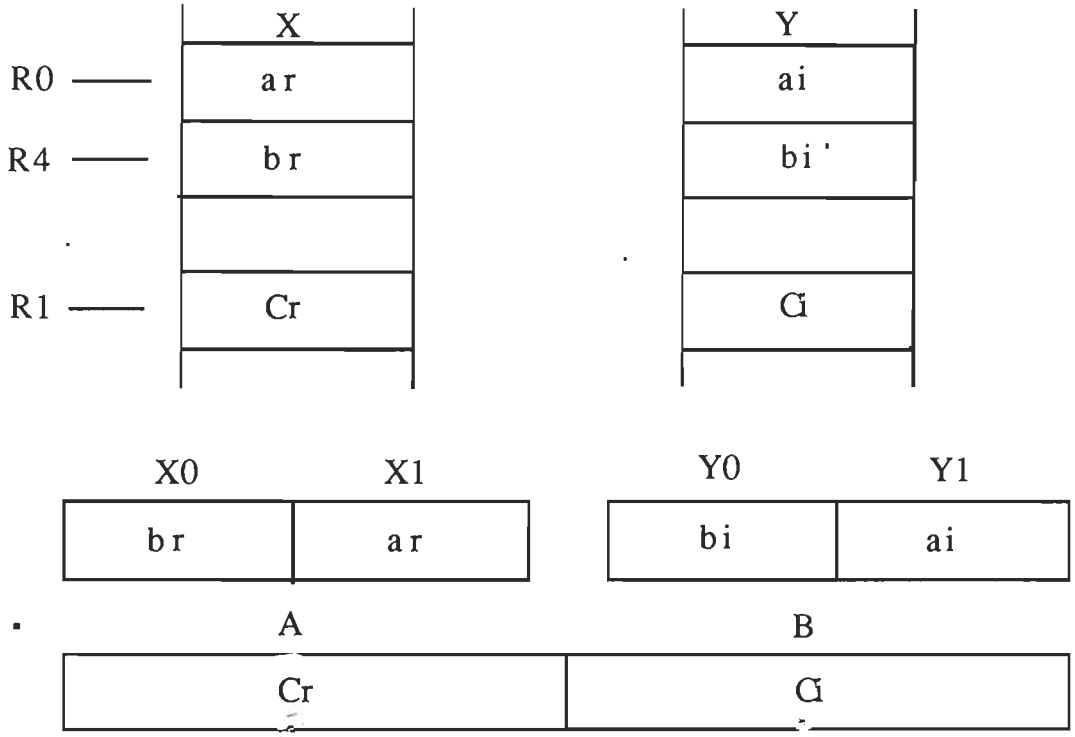
$$C_1 = K_1 \times K_2 = C_r + j\,C_i$$

$$C_r = a_r . b_r - a_i . b_i$$

$$C_i = a_r . b_i + a_i . b_r$$

On va utiliser les registres d'adresses suivantes :

- R0 pour pointer les adresses auxquelles se trouvent les données a_r et a_i dans les mémoires X et Y respectivement.
- R4 pour pointer les adresses auxquelles se trouvent les données b_r et b_i dans les mémoires X et Y respectivement .
- R1 pour pointer les adresses auxquelles on va mettre le résultat de multiplication C_r et C_i dans les mémoires X et Y.



Programme en langage assembleur:

```
move                X: (R0),X1      Y : (R4),Y0
mpy   Y0,X1,B       X: (R4),X0      Y : (R0),Y1
macr  X0,Y1,B
mpy   X0,X1,A                       B, Y: (R1)
macr -Y0,Y1,A
move                A ,X :(R1)
```

TABLEAU I

ARITHMETIC INSTRUCTIONS(Dst=56-bit A,B)

ABS	Absolute Value
ADC	Add Long with Carry
ADD	Add
ADDL	Shift Left and Add Accumulators
ADDR	Shift Right and Add Accumulators
ASL	Arithmetic Shift Accumulator Left
ASR	Arithmetic Shift Accumulator Right
CLR	Clear Accumulator
CMP	Compare
CMPM	Compare Magnitude
DIV	Divide Iteration △
MAC	Signed Multiply-Accumulate
MACR	Signed Multiply-Accumulate & Round
MPY	Signed Multiply
MPYR	Signed Multiply and Round
NEG	Negate
NORM	Normalize Accumulator Iteration △
RND	Round Accumulator
SBC	Subtract Long with Carry
SUB	Subtract
SUBL	Shift Left and Subtract Accumulators
SUBR	Shift Right and Subtract Accumulators
Tcc	Transfer Conditionally △
TFR	Transfer Data ALU Register
TST	Test Accumulator

LOOP INSTRUCTIONS

DO	Start Hardware Loop
ENDDO	End Current DO Loop

MOVE INSTRUCTIONS

LUA	Load Updated Address
MOVE	Move Data Registers ◁
MOVEC	Move Control Register
MOVEM	Move Program Memory
MOVEP	Move Peripheral Data

◁ Equivalent to Data ALU NOP with parallel data moves

BIT MANIPULATION INSTRUCTIONS (test mem)

BCLR	Bit Test and Clear
BSET	Bit Test and Set
BCHG	Bit Test and Change
BTST	Bit Test on Memory

Syntax: BCHG #n,X:<ea>
BCHG #n,Y:<ea>

JCLR	Jump if Bit Clear
JSET	Jump if Bit Set
JSCLR	Jump to Subroutine if Bit Clear
JSSET	Jump to Subroutine if Bit Set

Syntax: JSET #n,X:<ea>,xxxx
JSET #n,Y:<ea>,xxxx

PROGRAM CONTROL INSTRUCTIONS

Jcc	Jump Conditionally
JMP	Jump
JScc	Jump to Subroutine Conditionally
JSR	Jump to Subroutine
NOP	No Operation
REP	Repeat Next instruction
RESET	Reset On-Chip Peripheral Devices
RTI	Return from Interrupt
RTS	Return from Subroutine
STOP	Stop Instruction Processing ▽
SWI	Software Interrupt
WAIT	Wait for Interrupt ▽

LOGICAL INSTRUCTIONS (Dst=24-Bit A1,B1)

AND	Logical AND
ANDI	AND Immediate with Control Register
EOR	Logical Exclusive OR
LSL	Logical Shift Left
LSR	Logical Shift Right
NOT	Logical Complement
OR	Logical Inclusive OR
ORI	OR Immediate with Control Register △
ROL	Rotate Left
ROR	Rotate Right

PARALLEL DATA MOVE INSTRUCTIONS

• **Syntax:**

Opcode	Operands	Parallel Move: X Bus Data (Source 1 → Destination 1)	Parallel Move: Y Bus Data (Source 2 → Destination 2)
MACR	-X0, Y0, A	X:(R0) + ,X0	Y:(R4) - ,Y0

Specifies:

- Data ALU Operation
- Logical Operation
- Convergent Rounding

Supports:

- Condition Code Bits 0 through 5

Specifies:

- Up to two optional data transfers
- Two different addressing modes
- Address space qualifiers [X:, Y:, L:, P:, XY:]

Supports:

- Scaling
- Limiting
- Sign extension and least significant zero fill
- Duplicate sources
- Duplicate destinations not allowed

NCD02-0502

PARALLEL DATA MOVE OPERATIONS

Parallel Data Move

MACR -X0, Y0, A

1. Negate X0
2. 24 x 24 Bit Multiply X0 x Y0
3. 56 Bit Sum into Accumulator A
4. Convergently Round A
5. Update Condition Code Register

A,X:(R0)+N0

6. MOVE A to X Memory
7. Scale
8. Limit (If Necessary)
9. Calculate Next Address (R0)+ N0
10. Use Modifier Register M0 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

Y:(R4)-N4,B

11. MOVE Y Memory to B
12. Sign extend and Zero fill
13. Calculate Next Address (R4) - N4
14. Use Modifier Register M4 For
    - Linear Addressing
    - Modulo M
    - Bit Reverse

• This All Occurs In 97.5 nS.

NCD02-0503

VIII- APPLICATION DU PROCESSEUR NUMÉRIQUE DE SIGNAUX

En réponse aux besoins en performances nécessités par le traitement numérique en temps réel des signaux analogiques, les processeurs numériques de signaux trouvent plusieurs domaines d'application dans l'instrumentation, le contrôle des processus et autres. Toutefois on peut l'utiliser comme coprocesseur ou comme unité centrale selon la configuration de l'utilisation, comme le montre les figures 11 et 12.

Dans le domaine de filtrage numérique, le DSP56000 trouve son champ d'application pour tous les types des filtres à savoir :

- Les filtres à réponse impulsionnelle finie (FIR),
- Les filtres à réponse impulsionnelle infinie (IIR),
- Les filtres adaptatives, etc.

Les figures 13 et 14 montrent quelques configuration d'implantation des filtres à l'aide du processeur numérique de signal.

Quant aux performances du DSP56000/1 dans l'exécution des différents algorithmes de traitemant numérique de signaux, on peut les résumer au tableau suivant :

Performances du DSP56000/1

Algorithmes	Temps d'exécution
FIR avec décalage	0.1 µs par étage
FIR adaptatif réel	0.2 µs par étage
IIR biquad	0.4 µs
FFT 32 points complexes	67.3 µs
FFT 64 points complexes	147 µs
FFT 256 points complexes	713 µs
FFT 512 points complexes	2690 µs
FFT 1024 points complexes	5000 µs
Division	2.7 µs

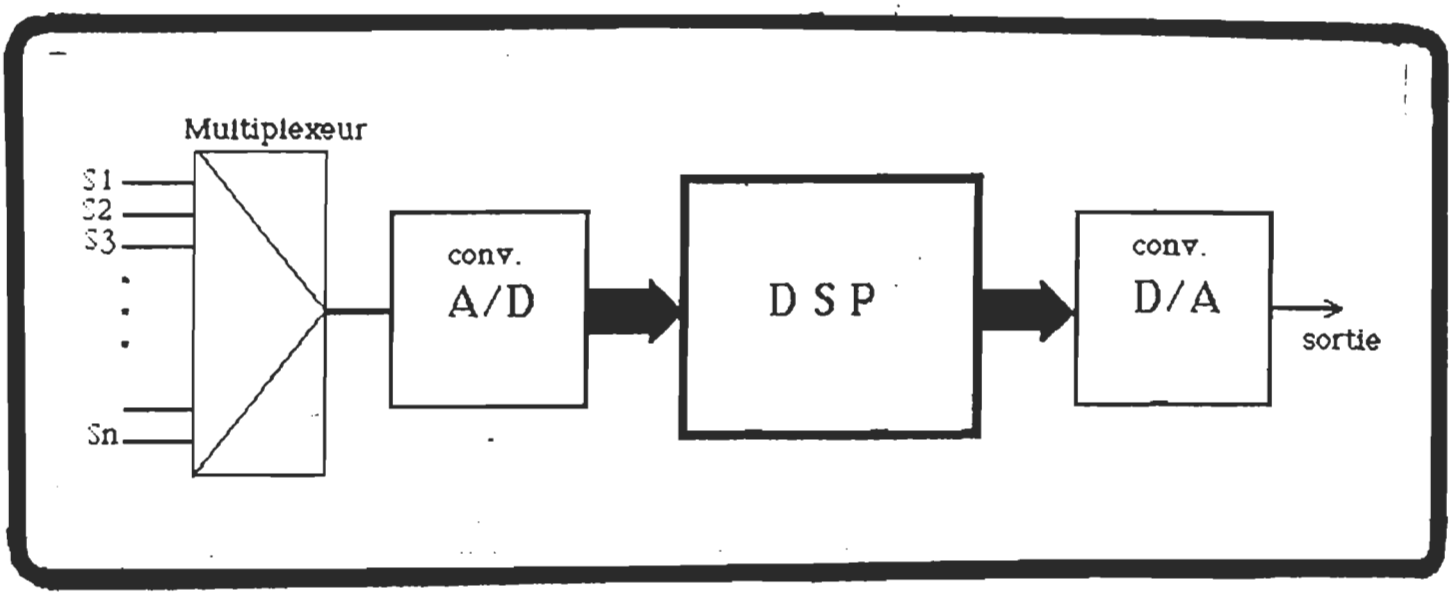


Fig.12 Configuration avec le DSP comme unité centrale.

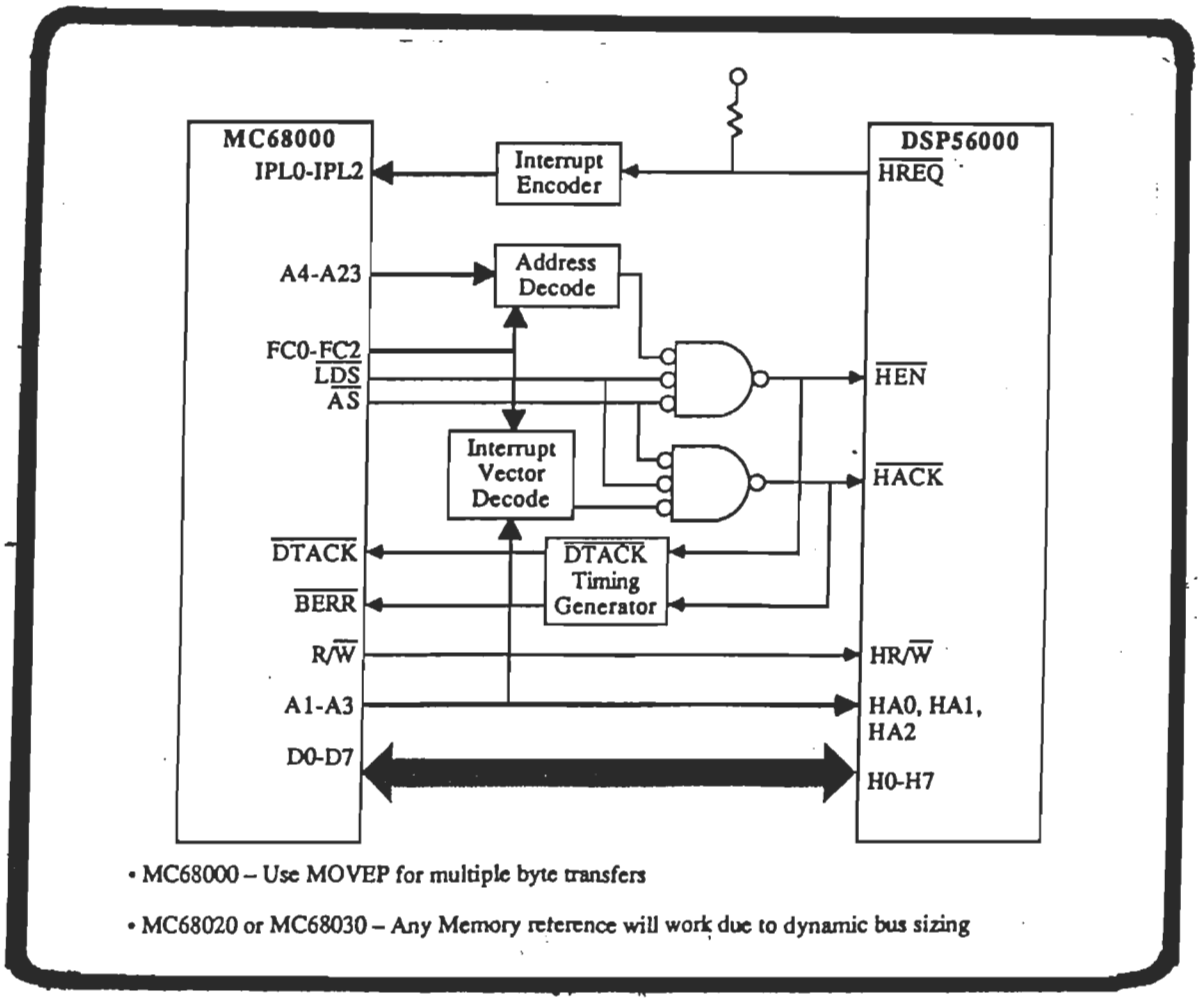


Fig.11 Configuration avec le DSP comme coprocesseur

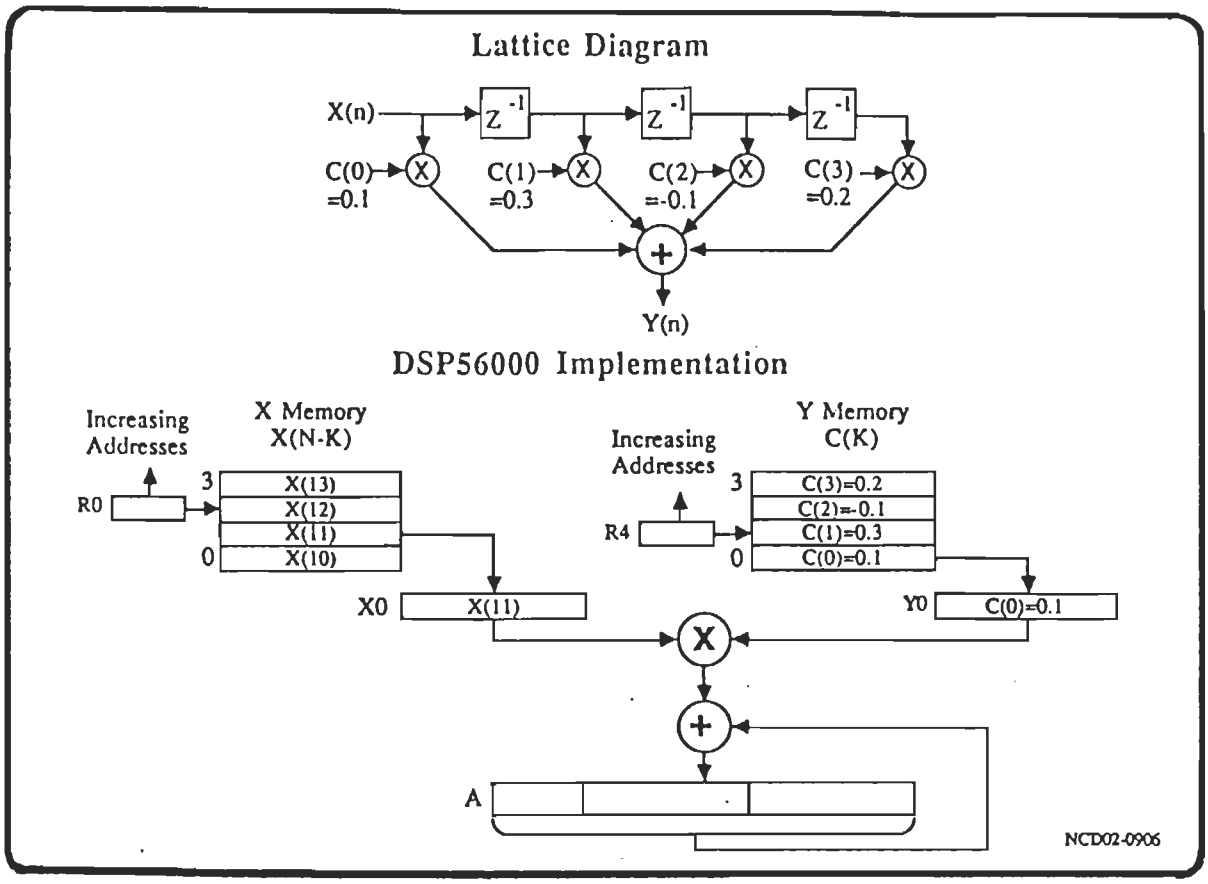


Fig.13 Configuration d'implantation d'un filtre FIR

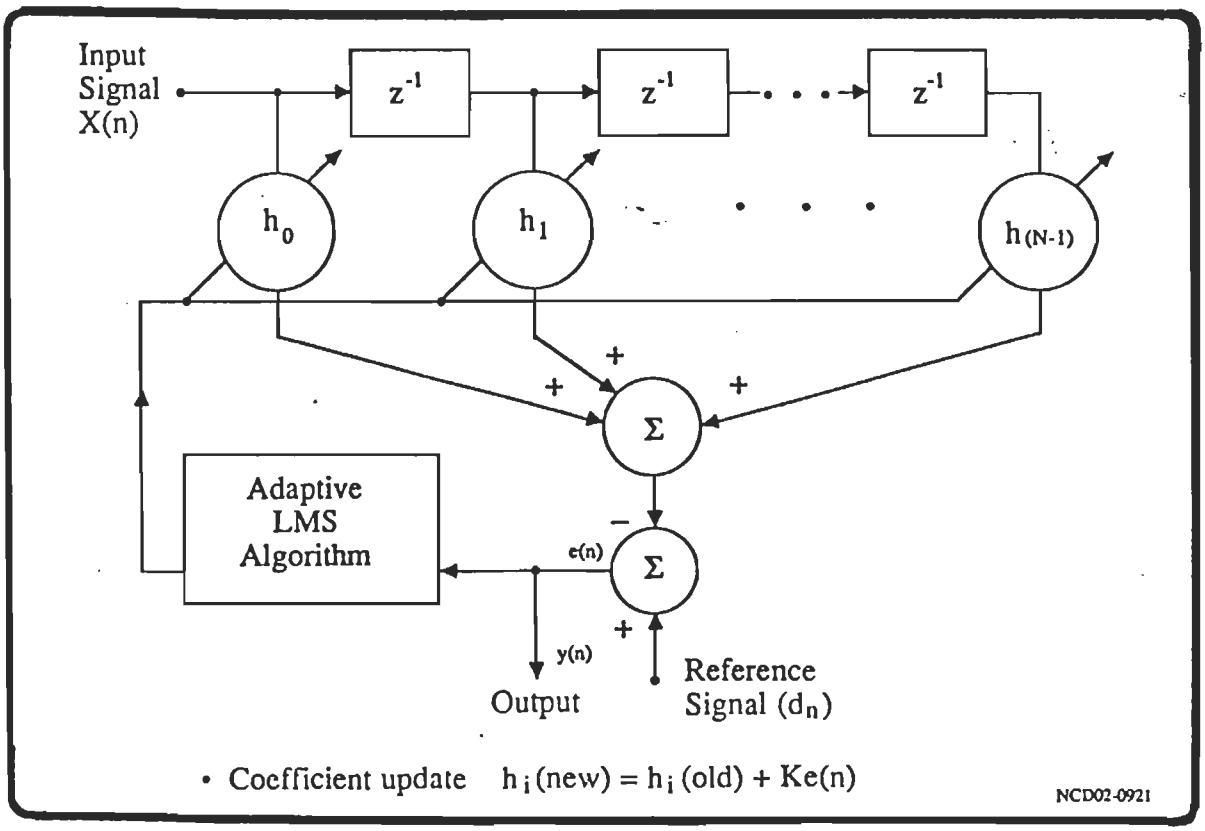


Fig.14 Configuration d'implantation d'un filtre adaptative.

IX- DÉVELOPPEMENT MATERIEL

Afin d'évaluer les performances de traitement du DSP56000/1, un outil de simulation matériel est élaboré. Il s'agit d'un système de développement et d'application dit DSP56000 ADS (" Application Development System "). Il comprend trois modules qui effectuent le développement des circuits pour la conception des systèmes qui traitent les signaux en temps réel :

- Module de développement et application (ADM) qui contient le DSP56001 et les circuits de contrôle.
- La carte d'interface HOST-BUS qui sert d'interface parallèle haute vitesse entre l'ordinateur et la carte d'émulation.
- Programme d'interface qui permet le dialogue entre l'utilisateur et la carte et il peut contrôler jusqu'a huit ADM.

La configuration matérielle est présenté aux figures 15 et 16.

Quelques caractéristiques matérielles

- Exploitation à haute vitesse 20.48 Mhz,
- 8K/32 Kmots de RAM disponible pour l'utilisateur,
- Pas d'alimentation externe,
- Connecteurs séparés pour accéder aux ports série et parallèle.

Quelques caractéristiques logicielles

- Simple/Multiple progression à l'intérieur du programme objet du DSP,
- Points d'arrêts conditionnel et inconditionnel,
- Chargement et sauvegarde de fichiers de/à la mémoire de l'ADM,
- Définition et exécution des macrocommandes,
- Visualisation des registres et des mémoires validés/non validés,
- Calculateur binaire/décimal/hexadécimal.

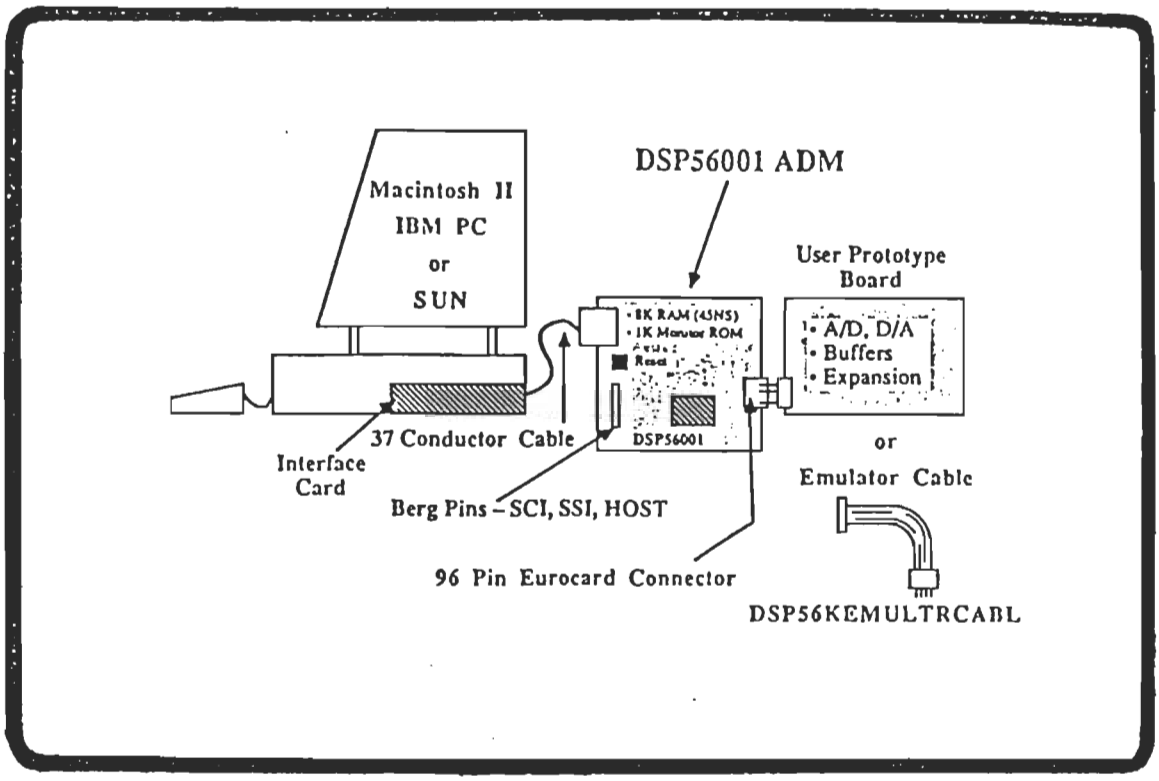


Fig.15:Configuration matérielle du système d'application ADS

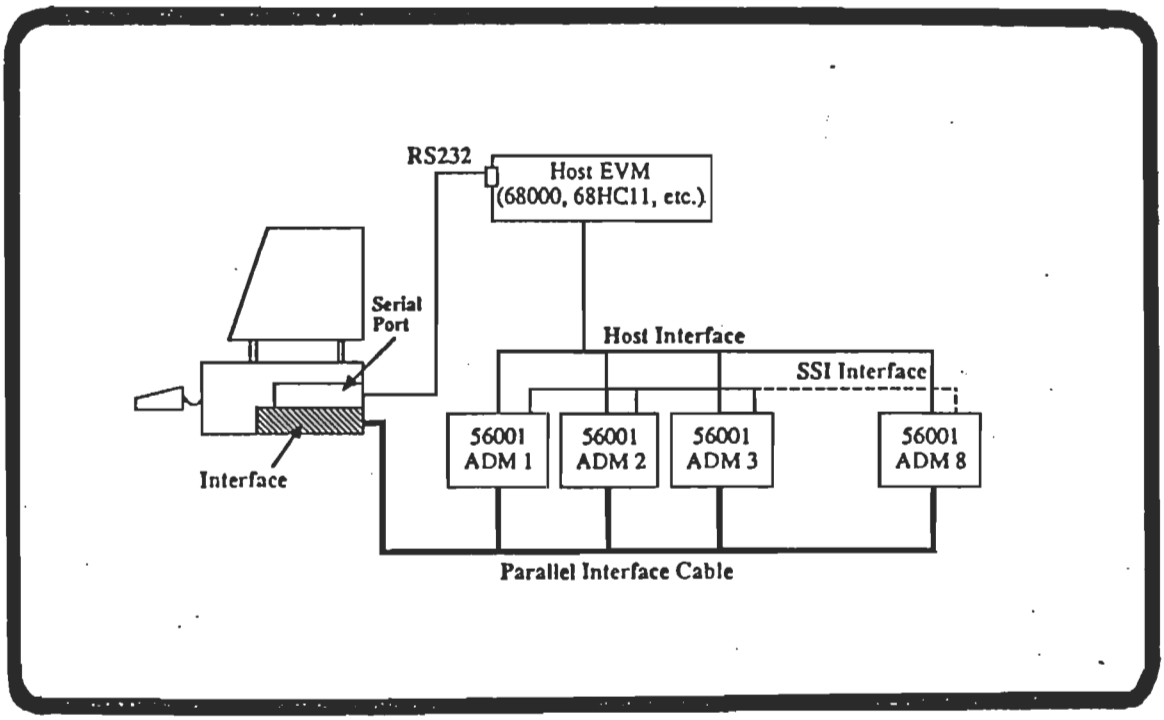


Fig.16:Possibilités de connexion de huit ADM avec le système

ANNEXE A-2

PROGRAMMES D'EXPÉRMENTATION
POUR LA CORRECTION DES ERREURS DE MULTIPLEXAGE

```
;************************************************
;***  PROGRAMME POUR LE TRAITEMENT DE QUATRE  ***
;***  SIGNAUX  SANS UTILISER LES FILTRES DE   ***
;***                CORRECTION                ***
;*** PROGRAMME ECRIT PAR "BEN SLIMA MOHAMED"  ***
;************************************************

;***** ADRESSE DU DEBUT DU PROGRAMME ******
        org     y:0
        org     p:$40

;METTRE LA CARTE ADS EN ETAT" force-break" AU LIEU DE
; "force-reset"

        movep   #0,x:$fffe
        movec   #0,sp
        movec   #0,sr

;METTRE LE PORT C EN FONCTIONNEMENT SCI/SSI

        move    #$0,a0
        movep   a0,x:$ffe1
        move    #$0001ff,a0
        movep   a0,x:$ffe1

;PROGRAMMATION DES REGISTRES DE CONTROLE CRA ET CRB DU PORT C

        move    #$004000,a0
        movep   a0,x:$ffec
        move    #$003200,a0
        movep   a0,x:$ffed

;PROGRAMMATION DU PORT B POUR LA SELECTION DU MULTIPLEXEUR

        move    #$0,a0
        move    a0,x:$ffe0
        move    #$3,a0
        move    a0,x:$ffe2

;********  DEBUT DU PROGRAMME  ************

        move    #$0,r0
        move    #-1,m0
        do      #300,_endp

    ;****  SIGNAL SUR CANAL #1 *****

        move    #$fffc,a0
        movep   a0,x:$ffe4
deb1    move    #$ffff00,x0
        jclr    #7,x:$ffee,deb1
        movep   x:$ffef,a
        and     x0,a
        move    a1,y:(r0)+
```

```
    ;**** SIGNAL SUR CANAL #2 ****

          move      #$fffd,a0
          movep     a0,x:$ffe4
deb2      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb2
          movep     x:$ffef,a
          and       x0,a
          move      a1,y:(r0)+

    ;**** SIGNAL SUR CANAL #3 ****

          move      #$fffe,a0
          movep     a0,x:$ffe4
deb3      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb3
          movep     x:$ffef,a
          and       x0,a
          move      a1,y:(r0)+

    ;**** SIGNAL SUR CANAL #4 ****

          move      #$ffff,a0
          movep     a0,x:$ffe4
deb4      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb4
          movep     x:$ffef,a
          and       x0,a
          move      a1,y:(r0)+
          move      a,x:$ffef
_endp
          end
```

```
;************************************************
;** PROGRAMME DE CORRECTION DE QUATRE SIGNAUX  **
;**  A TRAVERS LE MULTIPLEXEUR, LE FILTRE DE   **
;**       LAGRANGE UTILISE EST D'ORDRE 2       **
;**             ( R=2 , K=2 , L =0 )           **
;**  PROGRAMME ECRIT PAR " BEN SLIMA MOHAMED " **
;************************************************
          opt       nomd,nocex,nocm,nomex
          include   'fir_1coe'
          include   'fir_2coe'
          include   'fir_3coe'
;******* CONSTANTES ET MEMOIRE *************

ntaps     equ       3
          org       x:0
firdat0   dsm       ntaps
          org       x:$20
firdat1   dsm       ntaps
          org       x:$40
firdat2   dsm       ntaps

;******* PLACEMENT DES COEFFICIENTS DES FILTRES **********

          org       y:0

fir_1     filtcoef1

          org       y:$10

fir_2     filtcoef2

          org       y:$20

fir_3     filtcoef3

;*** ADRESSE DU DEBUT DU PROGRAMME***

          org       p:$40

;METTRE LA CARTE ADS EN "force-break" AU LIEU DE "force-reset"

          movep     #0,x:$fffe
          movec     #0,sp
          movec     #0,sr

;MISE A JOUR DES REGISTRES DES FILTRES

;********   FILTRE #1   **************

          move      #firdat0,r0
          move      #ntaps-1,m0
          move      #fir_1,r4
          move      #ntaps-1,m4
```

```
;********   FILTRE #2   **************

         move      #firdat1,r1
         move      #ntaps-1,m1
         move      #fir_2,r5
         move      #ntaps-1,m5

;********   FILTRE #3   **************

         move      #firdat2,r2
         move      #ntaps-1,m2
         move      #fir_3,r6
         move      #ntaps-1,m6

; METTRE LE PORT C EN FONCTIONNEMENT SCI/SSI

         move      #$0,a0
         movep     a0,x:$ffe1
         move      #$0001ff,a0
         movep     a0,x:$ffe1

; PROGRAMMATION DES REGISTRES DE CONTROLE CRA ET CRB DU PORT C

         move      #$004000,a0
         movep     a0,x:$ffec
         move      #$003200,a0
         movep     a0,x:$ffed

; PROGRAMMATION DU PORT B POUR LE MULTIPLEXEUR
; PB0 et PB1 EN SORTIE POUR SELECTIONNER LE CANAL SPECIFIE

         move      #$0,a0
         move      a0,x:$ffe0
         move      #$3,a0
         move      a0,x:$ffe2

; STOCKER LES RESULTATS DANS LA MEMOIRE Y EN UTILISANT
; LE REGISTRE D'ADRESSE (R7) COMME POINTEUR D'ADRESSE

         move      #$100,r7
         move      #-1,m7          ;adressage linéaire
         do        #400,_endp

;********* DEBUT DU PROGRAMME DE CORRECTION ************
;LE SIGNAL SUR LE CANAL 1 NE VA ETRE CORRIGE ,C'EST LUI
;QUI VA ETRE COMME REFERENCE POUR LES AUTRES SIGNAUX

         move      #$fffc,a0
         movep     a0,x:$ffe4
deb1     move      #$ffff00,x0
         jclr      #7,x:$ffee,deb1
         movep     x:$ffef,a
         and       x0,a
         move      a1,y:(r7)+
```

```
;**** CORRECTION DU SIGNAL SUR CANAL #2 *****************

          move      #$fffd,a0
          movep     a0,x:$ffe4
deb2      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb2
          movep     x:$ffef,a
          and       x0,a

;*********  DEBUT DE CORRECTION  ************************
          move      a1,x0
          clr       a          x0,x:(r0)+     y:(r4)+,y0
          rep       #ntaps-1
          mac       x0,y0,a    x:(r0)+,x0     y:(r4)+,y0
          macr      x0,y0,a    (r0)-
          move      a1,y:(r7)+

;**** CORRECTION DU SIGNAL SUR CANAL #3 *****************

          move      #$fffe,a0
          movep     a0,x:$ffe4
deb3      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb3
          movep     x:$ffef,a
          and       x0,a

;*********  DEBUT DE CORRECTION  ************************

          move      a1,x0
          clr       a          x0,x:(r1)+     y:(r5)+,y0
          rep       #ntaps-1
          mac       x0,y0,a    x:(r1)+,x0     y:(r5)+,y0
          macr      x0,y0,a    (r1)-
          move      a1,y:(r7)+

;**** CORRECTION DU SIGNAL SUR CANAL #4 *****************

          move      #$ffff,a0
          movep     a0,x:$ffe4
deb4      move      #$ffff00,x0
          jclr      #7,x:$ffee,deb4
          movep     x:$ffef,a
          and       x0,a

;*********  DEBUT DE CORRECTION  ************************
          move      a1,x0
          clr       a          x0,x:(r2)+     y:(r6)+,y0
          rep       #ntaps-1
          mac       x0,y0,a    x:(r2)+,x0     y:(r6)+,y0
          macr      x0,y0,a    (r2)-
          move      a1,y:(r7)+
          move      a,x:$ffef
_endp
          end
```

ANNEXE A-3

PROGRAMMES DE SIMULATIONS ET D'EXPÉRIMENTATIONS POUR L'AMÉLIORATION DE LA RÉSOLUTION DES MESURES SPECTROMÉTRIQUES

RECONSTITUTION DES SIGNAUX DE MESURE UTILISANT LA MÉTHODE DE RÉGULARISATION DE TIKHONOV (PROGRAMMES)

```
%*****************************************************************
% Programme de reconstitution des signaux utilisant la  méthode de régularisation
% de tikhonov. Cette méthode utilise les " informations a priori " sur le système
% de mesure à l'étude pour la correction et elle offre aussi une solution optimale
% pour la réduction de l'effet du bruit affectant le signal de mesure  en utilisant
% une critère d'erreur à minimiser dans un espace bien définie
%*****************************************************************

clear
N=input('NOMBRE DE POINTS DE DISCRETISATION    N  = ');
Sigma=input('AMPLITUDE DU BRUIT QUI AFFECTE LE SIGNAL      =');

h=2/N;              :pas de calcul

*****************************************************************
****   INTRODUCTION DES SIGNAUX D'ENTREES x(t) ET g(t)   ****
****    x(t) : signal d'entrée du système de mesure.          ****
****    g(t) : signal qui représente la réponse               ****
****           impulsionnelle du signal d'entrée.             ****
*****************************************************************

ti=-1.0+h/2;
t=h/2;
i=1;

  for k=1:N

            if ti>=-.50 & ti<.5
               g=6*exp(-80*(ti).^2);
            else
               g=0;
            end
```

```
            if t>=.5 & t<=1.5
              x=exp(-(t-.8)^2/.03)+exp(-(t-1.2)^2/.03);
              x=(x/.9550408-.052130913)*1.4*t;
            else
              x=0;
            end

    F(i)=x;
    T(i)=t;
    T1(i)=ti;
    H(i)=g;
    i=i+1;
    t=t+h;
    ti=ti+h;
    w(k)=2/h*sin(pi*(k-1)/N);
    w2(k)=1+w(k).^2;      :filtre pour le système à l'étude.
      end

*****************************************************************
******    INTRODUCTION DU BRUIT ADDITIF                  ******
*****************************************************************
      rand('normal');
      bruit1=rand(1,N);
      br=Sigma*bruit1;
      var=1/N*sum(br.^2);
      VARIANCE_Y=var

*****************************************************************
******    DETERMINATION DU SIGNAL DE SORTIE y(t)     ******
*****************************************************************

X=fft(F);                          %FFT du signal d'entrée
G=fft(H);                          %FFT du signal de convolution
XG=h*G.*X;
Yc=real(ifft(XG));
Yc1=fftshift(Yc);                  %signal de sortie sans bruit
Y1=Yc1+br;                         %signal de sortie avec bruit
Y=fft(Y1);                         %FFT du signal de sortie
Ym=real(Y).^2+imag(Y).^2;          %module de Y : | Y |²
Gm=real(G).^2+imag(G).^2;          %  | G |²
```

```
**********************************************************
******* ALGORITHME DE DETERMINATION DU PARAMETRE  *******
******* DE REGULARISATION PAR LA METHODE DE NEWTON *******
*******       D'APPROXIMATION SUCCESSIVE         *******
**********************************************************
**********************************************************
******  INTRODUCTION DES DONNEES QUI REPRESENTE  *******
******  LES INFORMATIONS A PRIORI SUR LE SYSTEME  *******
**********************************************************
a=input('VALEUR INITIALE DE ALPHA   =');
DeltaY=input('DELTA_Y =');
DeltaG=input('DELTA_G =');
eps=1e-20;
a1=1; a0=1;ic=0;ro=-1;

          ****************************************
          *****   DEBUT DE L'ALGORITHME    *****
          ****************************************

        while ro<0
F1=0;
F2=0;
F3=0;
     for j=1:N

BA=w2(j)./(Gm(j)+a*w2(j));
AB=1-a*BA;
F00=Ym(j)*BA;
F11=a^2*(F00*BA);
F22=F00*AB;
F33=2*F11*AB;
F1=F1+F11;
F2=F2+F22;
F3=F3+F33;
      end
F1=F1*h/N;
F2=F2*h/N;
F3=F3*h/N;
F21=sqrt(F2);
ro=F1-(DeltaY+DeltaG*F21)^2;
```

```
      if  ro<0
        a=2*a;
      end
                  end
ic=1;
    while abs(ro)>eps

F1=0;
F2=0;
F3=0;
      for j=1:N

BA=w2(j)./(Gm(j)+a*w2(j));
AB=1-a*BA;
F00=Ym(j).*BA;
F11=a^2*(F00*BA);
F22=F00*AB;
F33=2*F11*AB;
F1=F1+F11;
F2=F2+F22;
F3=F3+F33;
      end

F1=F1*h/N;
F2=F2*h/N;
F3=F3*h/N;
F21=sqrt(F2);
ro=F1-(DeltaY+DeltaG*F21)^2;

       if  abs(ro)>eps
          if ro  >=0
             a1  =a;
             r1  =ro;
          else
             a0  =a;
             r0  =ro;
              if ic==1
                 ic=2
              end
          end
```

```
                    if ic==i
      DR=-F3.*a-(DeltaY+DeltaG*F21)*DeltaG*F3/F21;
      DQ=-ro/DR;
      a=1./(1/a+DQ);
                    end
            if ic==2
             a=a0+(a1-a0)/2;
            end
       end
     end

      if abs(a)<1e-30
        error('ERREUR SUR LA VALEUR DE ALPHA ?')
      end
  end

*****************************************************************
****  RECONSTITUTION DU SIGNAL D'ENTREE A PARTIR DE     ****
****  L'INFORMATION SUR LE SIGNAL DE SORTIE , LE SIGNAL ****
****   DE CONVOLUTION ET LA VALEUR DU PARAMETRE ALPHA
****
*****************************************************************

Xe=Y.*conj(G);
Xe=Xe./(Gm+a*w2);
Xes=real(ifft(Xe))/h;
Xes1=fftshift(Xes);                    %signal reconstruite x@Ie@i(t)
var3=1/N*sum((Xes1-F).^2);
Erx=sqrt(var3)                         %erreur relative entre x(t) et
                              xe(t)


        ****************************************
        *** COURBES DES RESULTATS OBTENUES   ***
        ****************************************
plot(T1,H);grid;title('FONCTION G(t)');pause
plot(T,F);grid;title('FONCTION EXACTE DE X(t)');pause
plot(T,Y1,T,Yc1);grid;title('FONCTION Y(t):G(t)*X(t)');pause
plot(T,Xes1);grid;title('SIGNAL RECONSTRUITE X(t)')
end
```

EVALUATION DE L'ALGORITHME DE TIKHONOV

1°- PROGRAMME COFTRAN.M

```
**************************************************************
** Programme de détermination des coefficients de transmission   **
** des erreurs introduites par une variation brusque des donnés   **
** qui sont sujet à des erreurs qui sont dans notre cas les signaux **
** y(t) et g(t) et voir son influence sur l'erreur relative entre le **
** signal excate et le signal reconstruite xe(t).                 **
** Le coefficient de transmission des erreurs est définie par la    **
** relation :                                                     **
**                          ε[xe] - ε[x]                          **
**     Tn    = 100  ------------------------                      **
**                             ε[x]                               **
** où   n = 1,...,N.                                             **
**   ε[xe]:erreur relative suite à la variation du donnée         **
**   ε[x] :erreur exacte sans variation du donnée                 **
**************************************************************
```

```
N=input('NOMBRE DE POINTS        N  = ');
a=input('VALEUR INITIALE DE ALPHA   =');
Erx0=input('ERREUR EXACTE INITIALE          =');
h=2/N;
        ****************************************
        ****  INTRODUCTION DES DONNEES     ****
        ****************************************

ti=-1.0+h/2;
t=h/2;
i=1;
          for k=1:N
              if ti>=-.50 & ti<.5
                    g=6*exp(-80*(ti).^2);
              else
                    g=0;
              end
```

```
            if t>=.5 & t<=1.5
        x=exp(-(t-.8)^2/.03)+exp(-(t-1.2)^2/.03);
        x=(x/.9550408-.052130913)*1.4*t;
            else
        x=0;
            end
 F(i)=x;
 T(i)=t;
 T1(i)=ti;
 H(i)=g;
 i=i+1;
 t=t+h;
 ti=ti+h;
w(k)=2/h*sin(pi*(k-1)/N);
w2(k)=1+w(k).^2;
 end
        ***********************************************
        ****   INTRODUCTION D'ERREUR SUR g(t)   ****
        ***********************************************
            for J=1:N
Z=H;
Z(J)=1.01*H(J);
X=fft(F);
G=fft(Z);
XG=G.*X;
Yc=h*real(ifft(XG));
Yc1=fftshift(Yc);
Gm=real(G).^2+imag(G).^2;
        ****************************************
        ** RECONSTITUTION DU SIGNAL D'ENTREE  **
        ****************************************
Y=fft(Yc1);
Xe1=Y.*conj(G);
Xe=Xe1./(Gm+a*w2);
Xes=real(ifft(Xe))/h;
Xes1=fftshift(Xes);
var1=1/N*sum((Xes1-F).^2);
Erxn(J)=sqrt(var1);
Tn(J)=100*(Erxn(J)-Erx0)/Erx0;
            end
```

N.B : On refait la même chose pour y(t).

2°- PROGRAMME ERRFFT.M

```
%*********************************************************
% ** Programme d'évaluation de l'erreur introduite     **
% ** par l'effet de la représentation des nombres en   **
% ** mot binaire lors de l'exécution des opérations    **
% ** arithmétiques utilisés dans les programmes FFT.   **
% **   Dans ce cas il y aura coupure des mots ce qui   **
% ** peut augmenter les erreurs. Et en même temps on   **
% ** détermine le nombre de chiffre de mantisse qui    **
% **  sera suffisante pour ne pas introduire des       **
% **erreurs de ce genre en utilisant les coefficients **
% ** de transmission des erreurs introduites par la FFT*
% ** en se basant sur l'algorithme simplifié de       **
% ** " COOLEY-TUKEY ".                                 **
%*********************************************************

N=input('NOMBRE DE POINTS        N  = ');
a=input('VALEUR INITIALE DE ALPHA   =');
Sigma=input('AMPLITUDE DU BRUIT          =');
h=2/N;
          %****************************************
          %****  INTRODUCTION DES DONNEES   ****
          %****************************************
ti=-1.0+h/2;
t=h/2;
i=1;
              for k=1:N
               if ti>=-.50 & ti<.5
                    g=6*exp(-80*(ti).^2);
               else
                    g=0;
               end
               if t>=.5 & t<=1.5
              x=exp(-(t-.8)^2/.03)+exp(-(t-1.2)^2/.03);
              x=(x/.9550408-.052130913)*1.4*t;
                else
              x=0;
```

```
                end
 F(i)=x;
 T(i)=t;
 T1(i)=ti;
 H(i)=g;
 i=i+1;
 t=t+h;
 ti=ti+h;
w(k)=2/h*sin(pi*(k-1)/N);
w2(k)=1+w(k).^2;
                end
          ****************************************
          ***    INTRODUCTION DU BRUIT      ***
          ****************************************
rand('normal');
bruit1=rand(1,N);
br=Sigma*bruit1;
X=fft(F);
G=fft(H);
          ********************************************
          ** CALCUL DE L'ERREUR INTRODUITE PAR  **
          ** L'ALGORITHMME FFT DU SIGNAL g(t):dg **
          ********************************************
XG=real(H);
EXG=abs(XG);
YG=imag(H);
EYG=abs(YG);
M=log(N)/log(2);
N2=N;
    for K=1:M
        N1=N2;
        N2=N2/2;
        E=2*pi/N1;
        A=0;
            for J=1:N2
              C=cos(A);
              EC=abs(C);
              S=-sin(A);
              ES=abs(S);
              A=J*E;
```

```
                for I=J:N1:N
                  L=I+N2;
      EXTG=EXG(I)+EXG(L);
      XTG=XG(I)-XG(L);
      EXG(I)=EXG(I)+EXG(L);
      XG(I)=XG(I)+XG(L);
      EYTG=EYG(I)+EYG(L);
      YTG=YG(I)-YG(L);
      EYG(I)=EYG(I)+EYG(L);
      YG(I)=YG(I)+YG(L);
      EXG(L)=EXTG*abs(C)+EC*abs(XTG)+EYTG*abs(S)+ES*abs(YTG);
      XG(L)=XTG*C-YTG*S;
      EYG(L)=EXTG*abs(S)+ES*abs(XTG)+EYTG*abs(C)+EC*abs(YTG);
      YG(L)=XTG*S+YTG*C;
                end
            end
    end
dg=sqrt(EXG.^2+EYG.^2);

%DETERMINATION DU SIGNAL DE SORTIE :resultat de convolution

X0=G.*X;
Yc=h*real(ifft(X0));  :Signal de sortie y(t)
Yc2=fftshift(Yc);
Yc1=Yc2;

          ****************************************
          ** CALCUL DE L'ERREUR INTRODUITE PAR **
          **L'ALGORITHME FFT DU SIGNAL y(t):dy **
          ****************************************

X1=real(Yc1);
EX1=abs(X1);
Y1=imag(Yc1);
EY1=abs(Y1);
N2=N;

    for K=1:M
        N1=N2;
        N2=N2/2;
```

```
        E=2*pi/N1;
        A=0;
            for J=1:N2
              C=cos(A);
              EC=abs(C);
              S=-sin(A);
              ES=abs(S);
              A=J*E;

                    for I=J:N1:N
                      L=I+N2;
      EXT=EX1(I)+EX1(L);
      XT=X1(I)-X1(L);
      EX1(I)=EX1(I)+EX1(L);
      X1(I)=X1(I)+X1(L);
      EYT=EY1(I)+EY1(L);
      YT=Y1(I)-Y1(L);
      EY1(I)=EY1(I)+EY1(L);
      Y1(I)=Y1(I)+Y1(L);
      EX1(L)=EXT*abs(C)+EC*abs(XT)+EYT*abs(S)+ES*abs(YT);
      X1(L)=XT*C-YT*S;
      EY1(L)=EXT*abs(S)+ES*abs(XT)+EYT*abs(C)+EC*abs(YT);
      Y1(L)=XT*S+YT*C;
                  end
            end
    end
dy=sqrt(EX1.^2+EY1.^2);


********** RECONSTRUCTION DU SIGNAL D'ENTREE **********

Y=fft(Yc1);
Ym=real(Y).^2+imag(Y).^2;
Gm=real(G).^2+imag(G).^2;
Xe=Y.*conj(G);
Xe=Xe./(Gm+a*w2);
Xes=real(ifft(Xe))/h;
Xes=fftshift(Xes);
var=1/N*sum((Xes-F).^2);
Erx0=sqrt(var);
```

```
**********************************************************
** CALCUL DE L'ERREUR SUR LE SIGNAL RECONSTRUITE :dx **
**********************************************************

  for J=1:N
dw2(J)=abs(w2(J));
d1(J)=abs(Y(J)).*dg(J)+abs(conj(G(J))).*dy(J);
d2(J)=2*abs(G(J)).*dg(J)+a*dw2(J)+a*w2(J);
d3(J)=(Gm(J)+a*w2(J));
dx1(J)=(d1(J).*d3(J)+abs(Y(J)).*abs(G(J)).*d2(J));
dx(J)=dx1(J)./(d3(J).*d3(J));
  end
dx=fftshift(dx);

% détermination du nombre de chiffre de mantisse

m=-log(10*Erx0/(5*max(dx)))/log(10);
end
```

```
;*****************************************************************
; Programme de reconstitution de mesurande en utilisant la methode de
;deconvolution spectrale avec régularisation de Tikhonov
;*****************************************************************

            include   'a:sincos'
            include   'a:bitrev'
            include   'a:fft2'
            include   'a:datax'
            include   'a:datag'
            include   'a:mulcpx'
            include   'a:bruit'
            include   'a:datxi'
            include   'a:datgi'
            include   'a:omega'

reset       equ       0
start       equ       $100
points      equ       128
datax       equ       0
datag       equ       $100
coef        equ       128
datayb      equ       $180
olddatax    equ       $200
olddatag    equ       $280
datayr      equ       $300
noise       equ       $380
datay       equ       $400
omega       equ       $480
proyg       equ       $500
modg2       equ       $580
sigrex      equ       $700

; points        : nombre de points
; coef          : adresse de base de la table sincos
; olddatax      : adresse des données avant bit inverse de x(t)
; olddatag      : adresse des données avant bit inverse de g(t)
; datax         : adresse des données aprés bit inverse de x(t)
; datag         : adresse des données aprés bit inverse de g(t)
; datay         : adresse des données pour le spectre Y(f)
```

```
; datayr        : adresse des données de la transformee de fourier inverse du
;                 spectre Y(f)
; datayb        : adresse des données (y(t) +bruit) aprés bit inverse
; omega         : adresse des données du filtre w
; proyg         : adresse des données du produit Y(f).conj(G(f))
; modg2         : adresse des données du module de G au carré
; sigrex        : adresse des données du signal reconstitue

              org     x:$200

olddataxr     datx                    ; signal d'entrée  x(t)

              org     y:$200

olddataxi     dati

              org     x:$280

olddatagr     datg                    ; réponse impulsionnelle g(t)

              org     y:$280

olddatagi     datgi

              org     y:$380

bruit         noise

              org     y:$480

omegad        omega1

              org     x:$600

sigma         dc      .0001
alpha         dc      .3555556        ; alpha = alpha/.5625
cste          dc      6.1035E-005

              sincos  points,coef
```

```
            org     p:reset
            jmp     start
            org     p:start

            bitrev  points,datax,olddatax
            nop

            bitrev  points,datag,olddatag
            nop
;
;*************************************************
;           FFT du signal x(t)
;*************************************************

            fft2    points,datax,coef,olddatax  ;X(f)=FFT(x(t))
            nop

            move    #datax,r0
            do      #points,_end_mpy
            move    x:(r0),a
            move    y:(r0),b
            rep     #2
            lsl     a
            nop
            rep     #2
            lsl     b
            nop
            move    a,x:(r0)
            move    b,y:(r0)
            move    (r0)+
_end_mpy
            nop
;
;*************************************************
;         FFT du signal g(t)
;*************************************************

            fft2    points,datag,coef,olddatag
            nop
```

```
            move      #datag,r0
            do        #points,_end_mpy1
            move      x:(r0),a
            move      y:(r0),b
            rep       #3
            lsl       a
            nop
            rep       #3
            lsl       b
            nop
            move      a,x:(r0)
            move      b,y:(r0)
            move      (r0)+
_end_mpy1
            nop
;
;*************************************************************
;           Y(f) = X(f).G(f)
;*************************************************************

            mulcpx    points,datax,datag,datay
            nop

;*************************************************************
;           IFFT du spectre Y(f)
;*************************************************************

            move      #datay,r0

            do        #points,_end_conjy
            move      y:(r0),b
            neg       b
            move      b,y:(r0)+
_end_conjy
            nop
            bitrev    points,datayr,datay
            nop
            fft2      points,datayr,coef,datay
            nop
;
```

```
;*************************************************************
;          FFT du signal ( y(t)+bruit )
;*************************************************************

            move      #datayr,r0
            do        #points,_end_muly

            move      x:(r0),a
            rep       #4
            lsl       a
            move      a,x:(r0)+
_end_muly
            move      #datayr,r0
            move      #noise,r1
            move      #sigma,r4

            do        #points,_end_noise

            move      y:(r1)+,y0      x:(r4),x0
            mpyr      x0,y0,a         x:(r0),b
            addr      b,a
            move      a,x:(r0)+

_end_noise
            nop
            bitrev    points,datayb,datayr
            nop
            fft2      points,datayb,coef,datayr
            nop

            move      #datayb,r0
            do        #points,_end_mpy3
            move      x:(r0),a
            move      y:(r0),b
            rep       #3
            lsl       a
            nop
            rep       #3
            lsl       b
            nop
```

```
            move    a,x:(r0)
            move    b,y:(r0)
            move    (r0)+
_end_mpy3
            nop

;*********************************************************
;   Calcul du spectre du signal reconstitué  :
;
;    X (f) =  Y(f).G*(f) / ( |  G(f)  | ² + a( 1 + w² )
;*********************************************************
;
;*********************************************************
;      calcul du conjugué et du module de G
;*********************************************************
            move    #datag,r0
            move    #modg2,r1

            do      #points,_end_mod

            move    x:(r0),x0
            mpyr    x0,x0,a     y:(r0),y0
            mpyr    y0,y0,b
            add     b,a         (r0)+
            move    a,x:(r1)+
_end_mod
            nop
;
;*********************************************************
;   calcul du terme   Y(f) . conj(G(f))
;*********************************************************

            move    #datag,r0

            do      #points,_end_numx

            move    y:(r0),b
            neg     b
            move    b,y:(r0)+
_end_numx
```

```
            nop
            mulcpx    points,datayb,datag,proyg
            nop

;*****************************************************
;  calcul du terme  | G |² + α( 1 + w² )
;*****************************************************

            move      #omega,r0
            move      #cste,r4
            move      #alpha,r1
            move      #modg2,r2

            do        #points,_end_omega

            move      x:(r4),b        y:(r0)+,y0
            mpyr      y0,y0,a
            add       b,a             x:(r1),x0
            move      a,y0            x:(r2),b
            macr      x0,y0,b
            move      b,x:(r2)+

_end_omega

;*****************************************************
;      calcul de la FFT du signal reconstitué
;*****************************************************

            move      #modg2,r0
            move      #proyg,r4
            do        #points,_end_divr
            move      x:(r4),a
            move      x:(r0)+,x0

;division de la partie reelle

            jsr       divis
            move      x1,x:(r4)+
            nop
_end_divr
```

```
            move      #modg2,r0
            move      #proyg,r4

            do        #points,_end_divi

            move      y:(r4),a      x:(r0)+,x0

;division de la partie imaginaire

            jsr       divis
            move      x1,y:(r4)+
            nop
_end_divi
            nop
;
;*************************************************************
;*            calcul de IFFT( x(t) )                 *
;*************************************************************

            move      #proyg,r0

            do        #points,_end_conjx
            move      y:(r0),b
            neg       b
            move      b,y:(r0)+
_end_conjx
            nop
            bitrev    points,sigrex,proyg
            nop
            fft2      points,sigrex,coef,proyg
            nop
            move      #sigrex,r0

            do        #points,_end_end
            move      x:(r0),a
            rep       #5
            lsl       a
            move      a,x:(r0)+
_end_end
            nop
```

```
;*********************************************************
;       calcul de l'erreur de reconstitution
;*********************************************************

            move      #sigrex,r0
            move      #olddatax,r4
            clr       a
            clr       b

            do        #points,_endf
            move                    x:(r0)+,x0
            move                    x:(r4)+,a
            sub       x0,a
            move      a,y0
            macr      y0,y0,b
_endf
            rep       #$a
            lsl       b
            move      b,x:$605
            nop
;*********************************************************
;         Sous programme de division
;*********************************************************

divis       abs       a             a,b
            eor       x0,b          b,x:$0
            and       #$fe,ccr
            rep       #$18
            div       x0,a
            tfr       a,b
            jpl       savequo
            neg       b
savequo     tfr       x0,b          b0,x1
            abs       b
            add       a,b
            jclr      #13,x:$0,done
            move      #$0,b0
            neg       b
done        rts
            end
```

```
                    ******************************
                    ********  FFT2.ASM  ********
                    ******************************
;*************************************************************
;
;                       DECIMATION IN TIME  FFT
;
;   The outputs of all of the FFT butterflies are down-scaled
;   by a factor of two.
;*************************************************************

fft2        macro     points,data,coef,olddata

; passes  = number of passes = log2 (number of points)
; points  = number of points
; coef    = base address of sine/cosine table
; olddata = starting address of input data before bit reverse
; data    = starting address of input data after bit reverse
;

;*************************************************************
;                          FFT INITIALISATION
;*************************************************************

            move      #1,n0                     ; n0=group offset
            move      #points/2,n2              ; n2=group per pass

            move      #-1,m0                    ; m0=m1=m2=m4=m5=m6
            move      m0,m1                     ; linear addressing
            move      m0,m2
            move      m0,m4
            move      m0,m5
            move      m0,m6

;*************************************************************
;                             FFT PASSES
;*************************************************************

            do        #@cvi(@log(points)/@log(2)+0.5),_end_pass

            move      #data,r0          ; r0=ar,ai input pointer
            move      r0,r4             ; r4=ar',ai' output pointer
            move      #coef,r6          ; r6=wr,wi input pointer
            nop
            lua       (r0)+n0,r1        ; r1=br,bi input pointer
            nop
            lua       (r1)-,r5          ; r5=br',bi' output pointer

            move      n0,n1             ; n0 = n1 = n4 = n5
            move      n0,n4             ; group offset
            move      n0,n5
            move      n2,n6             ; n6 = coefficient offset
```

```
;*****************************************************************
;                              FFT GROUP
;*****************************************************************

            ori       #$4,mr
            do        n2,_end_grp
            move                          x:(r5),a          y:(r0),b
            lsl       a

;*****************************************************************
;                            FFT BUTTERFLY
;*****************************************************************

            do        n0,_end_bfy

            move                          x:(r1),x1         y:(r6),y0
            mac       x1,y0,b             x:(r6)+n6,x0      y:(r1)+,y1
            macr      -x0,y1,b            a,x:(r5)+         y:(r0),a
            subl      b,a                 x:(r0),b          b,y:(r4)
            mac       -x0,x1,b            x:(r0)+,a         a,y:(r5)
            macr      -y0,y1,b            x:(r1),x1
            subl      b,a                 b,x:(r4)+         y:(r0),b

_end_bfy

            move      #coef,r6
            move                          a,x:(r5)+n5       y:(r1)+n1,y1
            move                          x:(r0)+n0,x1      y:(r4)+n4,y1

_end_grp

            andi      #$fb,mr
            move                          n2,b1
            lsr       b                   n0,a1
            lsl       a                   b1,n2
            move      a1,n0

_end_pass

            endm

                  ********************************
                  ********  BITREV.ASM  ********
                  ********************************

;*****************************************************************
;           Storing input data in Bit-Reversed Order
;*****************************************************************

bitrev      macro         points,data,olddata

; points  = Number of points
; olddata = Input data in normal order
; data    = Input data in Bit-reversed order
```

```
            move        #olddata,r1         ; initialize the
            move        r1,r6               ; pointers
            move        #data,r0
            move        r0,r5
            move        #0,m1               ; m1 and m6 = bit
            move        m1,m6               ; reverse
            move        #-1,m0              ; m0 and m5 = linear
            move        m0,m5
            move        #points/2,n1
            move        n1,n6

            do          #points,_end_reverse
            move                    x:(r1)+n1,a     y:(r6)+n6,b
            move                    a,x:(r0)+       b,y:(r5)+

_end_reverse
            endm

                  ********************************
                  ********  MULCPX.ASM  ********
                  ********************************
;*************************************************************
;             Multiplication of two complexes data
;      data1 = ar + j ai
;                                data3 = data1 . data2
;      data2 = br + j bi
;
;      ar and br ( real numbers ) are stored in X memory
;      ai and bi ( imaginary numbers ) are stored in Y memory
;*************************************************************

mulcpx      macro       points,data1,data2,data3

; points  = Number of points
; data1   = Starting address of data1
; data2   = Starting address of data2
; data3   = Starting address of the result of the multiplication

            move        #data1,r0           ; initialize the
            move        #data2,r4           ; pointers
            move        #data3,r1
            move        #-1,m0              ; m0 = m1 = m4
            move        m0,m4               ; linear addressing
            move        m0,m1

            do          #points,_end_mcpx
            move                    x:(r0),x1       y:(r4),y0
            mpy         y0,x1,b     x:(r4)+,x0      y:(r0)+,y1
            macr        x0,y1,b
            mpy         x0,x1,a                     b,y:(r1)
            macr        -y0,y1,a
            move                    a,x:(r1)+
_end_mcpx
            endm
```

```
                    ********************************
                    ********  SINCOS.ASM  ********
                    ********************************

;*****************************************************************
;               Sine-Cosine Table Generator for FFTs
;*****************************************************************

; This program originally available on the Motorola DSP bulletin
; board.
; It is provided under a DISCLAMER OF WARRANTY available from
; Motorola DSP Operation, 6501 Wm. Cannon Drive W., Austin, Tx.,
; 78735.
;
; Last Update 25 Nov 86   Version 1.2
;
sincos  macro   points,coef
sincos  ident   1,2
;
;       sincos  -   macro to generate sine and cosine coefficient
;                   lookup tables for Decimation in Time FFT
;                   twiddle factors.
;
;       points  -   number of points (2 - 32768, power of 2)
;       coef    -   base address of sine/cosine table
;                   negative cosine value in X memory
;                   negative sine value in Y memory
;
; Latest revision - 25-Nov-86
;

pi      equ     3.141592654
freq    equ     2.0*pi/@cvf(points)

        org     x:coef
count   set     0
        dup     points/2
        dc      -@cos(@cvf(count)*freq)
count   set     count+1
        endm

        org     y:coef
count   set     0
        dup     points/2
        dc      -@sin(@cvf(count)*freq)
count   set     count+1
        endm

        endm    ;end of sincos macro
```

ANNEXE A-4

PROGRAMMES DE SIMULATIONS
POUR L'ÉTALONNAGE STATIQUE DU SYSTEME DE MESURE
PAR LES FONCTIONS SPLINE

SPLINE CUBIQUE

La fonction d'interpolation du spline cubique est définie par

$$Yn(x) = An(x-Xn)^3 + Bn(x-Xn)^2 + Cn(x-Xn) + Dn$$

ou An, Bn, Cn et Dn sont les coefficients du spline qui interpole la séquence yi=f(xi) pour i=1,...,N.

Les sous_programmes utilisés pour l'évaluation du spline cubique sont les suivants:

*Intspl3 : Il détermine les coefficients An, Bn, Cn et Dn pour chaque sub-intervalle .

*Valspl3 : Il détermine la valeur du spline pour X donnée ainsi que la valeur de sa première dérivée au meme point.

*Valspl3A : Il determine la valeur du spline en utilisant sa deuxième dérivée.

*Optspl3 : Il détermine les coefficients du spline An, Bn, Cn et Dn en minimisant la norme de sa troisième derivée dans l'espace L2.

*Eintspl3 : Le programme d'évaluation du spline cubique INTSPL3 pour la calibration du système de mesure pour l'analyse ultrasonore des solutions.

*Eoptspl3 : Le programme d'évaluation du spline cubique OPTSPL3 pour le calibration du système de mesure.

1-/ Sous-programme INTSPL3.M

```
function output=intspl3(X,Y)
 ************************************************************
 * fonction d'interpolation utilisant le "spline cubique"  *
 * de la forme :                                            *
 *     Yn(x) = An (x-Xn)^3 + Bn (x-Xn)^2 + Cn (x-Xn) +Dn    *
 * cette fonction est définie pour chaque sub-intervalle    *
 * de la fonction globale  yi=f(xi)  i=1,...,N              *
 ************************************************************
 ************************************************************
 * programme de détermination des coefficients An,Bn,Cn et *
 * Dn du spline cubique pour chaque noeuds (X(n),Y(n)) avec*
 * n=1,...,N en utilisant comme conditions initiales la    *
 * connaissance de la première dérivée (dy/dx) pour les    *
 * limites x=X(1) et x=X(N) notée Y11 et Y1N respectivement*
 ************************************************************

N=length(X);

% détermination de la valeur de (dy/dx) pour x=X1 et x=XN en utilisant la %
% formule d'interpolation de lagrange pour (X1,Y1);(X2,Y2);(X3,Y3)et (X4,Y4)
% détermination de Y11=(dy/dx) pour x=X(1);

dX1=X(2)-X(1);dX2=X(3)-X(1);dX3=X(4)-X(1);
K1=-1/dX1-1/dX2-1/dX3;
K2=dX2*dX3/dX1/(X(3)-X(2))/(X(4)-X(2));
K3=dX1*dX3/dX2/(X(2)-X(3))/(X(4)-X(3));
K4=dX1*dX2/dX3/(X(2)-X(4))/(X(3)-X(4));
Y11=K1*Y(1)+K2*Y(2)+K3*Y(3)+K4*Y(4);

% détermination de Y1N=(dy/dx) pour x=X(N);

dX1N=X(N-1)-X(N);dX2N=X(N-2)-X(N);dX3N=X(N-3)-X(N);
K1N=-1/dX1N-1/dX2N-1/dX3N;
K2N=dX2N*dX3N/dX1N/(X(N-2)-X(N-1))/(X(N-3)-X(N-1));
K3N=dX1N*dX3N/dX2N/(X(N-1)-X(N-2))/(X(N-3)-X(N-2));
K4N=dX1N*dX2N/dX3N/(X(N-1)-X(N-3))/(X(N-2)-X(N-3));
Y1N=K1N*Y(N)+K2N*Y(N-1)+K3N*Y(N-2)+K4N*Y(N-3);
```

```
% détermination des coefficients An,Bn,Cn et Dn du spline cubique qui interpole
% les series de points (X(n);Y(n) pour n=1,....,N selon la procédure définie dans
% le chapitre III-3.
*** calcul de Sn et de Delta(Xn) ****

        for k=1:N-1

     DT(k)=X(k+1)-X(k);
     S(k)=(Y(k+1)-Y(k))/DT(k);
     D(k)=Y(k);                          % le coefficient  Dn
       end
*** calcul des termes  Beta(n) et Gamma(n) ***

BT(1)=0;GM(1)=Y11;BT(2)=1;GM(2)=0;

          for j=3:N

   K=DT(j-1)/DT(j-2);
   P=-2*(1+K);
   R=3*(S(j-1)+K*S(j-2));
   BT(j)=P*BT(j-1)-K*BT(j-2);          % Betta (n)
   GM(j)=P*GM(j-1)-K*GM(j-2)+R;        % Gamma (n)
         end

** détermination du coefficient C2 en utilisant C(N)=Y1N **

              C2=(Y1N-GM(N))/BT(N);
*** détermination des coefficients An ; Bn et Cn

C(1)=Y11;
        for k=1:(N-2)

  C(k+1)=BT(k+1)*C2+GM(k+1);
  B(k)=(3*S(k)-C(k+1)-2*C(k))./DT(k);
  A(k)=(C(k+1)+C(k)-2*S(k))./(DT(k)).^2;
       end
  B(N-1)=(3*S(N-1)-Y1N-2*C(N-1))/DT(N-1);
  A(N-1)=(Y1N+C(N-1)-2*S(N-1))/(DT(N-1))^2;
output=[A' B' C' D'];
end
```

2-/ Sous-programme VALSPL3A.M

```
function [VY,VY1]=valspl3a(X,A,B,C,D,VX)
*****************************************************************
* détermination de la valeur du spline parabolique VY et sa   *
*première derivée VY1 définies par les abcisses des noeuds X *
* et les coefficients A,B et C.                                 *
*****************************************************************
N=length(X);
IL=1;IH=N;
                while IH>IL+1
  I=fix((IL+IH)/2);
     if X(I)>VX
          IH=I;
     else
          IL=I;
     end
      if IL==IH
           if IL==N
              IL=IL-1;
           else
              IH=IH+1;
           end
       end
         H=X(IH)-X(IL);
         DX=VX-X(IL);
         VY=((A(IL)*DX+B(IL))*DX+C(IL))*DX+D(IL);
         VY1=(3*A(IL)*DX+B(IL))*DX+C(IL);

                end
```

3-/ Sous-programme DER2SPL3.M

```
function  [Y2,Y1n]=der2spl3(X,Y,Y21)
 ***********************************************************
 * programme de détermination du vecteur Y2 qui représente *
 * les valeurs du deuxième dérivée du spline cubique.      *
 *   Y11 : la valeur du premier dérivée au point X1.       *
 *   Y21 : la valeur du second dérivée au point  X1.       *
 ***********************************************************
```

```
N=length(X);

détermination de la valeur de (dy/dx) pour x=X1 utilisant la
formule d'interpolation de lagrange pour (X1,Y1),(X2,Y2);
(X3,Y3)et (X4,Y4)

dX1=X(2)-X(1);dX2=X(3)-X(1);dX3=X(4)-X(1);
K1=-1/dX1-1/dX2-1/dX3;
K2=dX2*dX3/dX1/(X(3)-X(2))/(X(4)-X(2));
K3=dX1*dX3/dX2/(X(2)-X(3))/(X(4)-X(3));
K4=dX1*dX2/dX3/(X(2)-X(4))/(X(3)-X(4));
Y11=K1*Y(1)+K2*Y(2)+K3*Y(3)+K4*Y(4);

**** détermination des termes S(n) et DT(n) ****


        for I=2:N
    DT(I)=X(I)-X(I-1);
    S(I)=(Y(I)-Y(I-1))/DT(I);
        end

Y2(1)=Y21; Y1n=Y11;

        for I=2:N
  Y2(I)=6*(S(I)-Y1n)/DT(I)-2*Y2(I-1);
  Y1n=Y1n+(Y2(I)+Y2(I-1))*DT(I)/2;
        end
end
```

4-/ Sous-programme SPL3_Y2.M

```
function  output=spl3_y2(X,Y)

*************************************************************
* programme de determination du vecteur Y2 qui represente *
* les valeurs du second derivee du spline cubique.        *
*   Y11 : la valeur du premier derivee au point X1.       *
*   Y21 : la valeur du second derivee au point  X1.       *
*************************************************************
N=length(X);
```

```
**** détermination de Y1N=(dy/dx) pour x=X(N) ****

dX1N=X(N-1)-X(N);dX2N=X(N-2)-X(N);dX3N=X(N-3)-X(N);
K1N=-1/dX1N-1/dX2N-1/dX3N;
K2N=dX2N*dX3N/dX1N/(X(N-2)-X(N-1))/(X(N-3)-X(N-1));
K3N=dX1N*dX3N/dX2N/(X(N-1)-X(N-2))/(X(N-3)-X(N-2));
K4N=dX1N*dX2N/dX3N/(X(N-1)-X(N-3))/(X(N-2)-X(N-3));
Y1N=K1N*Y(N)+K2N*Y(N-1)+K3N*Y(N-2)+K4N*Y(N-3);
**** détermination du vecteur Y2 *****

[z1,z2]=der2spl3(X,Y,0);
[z3,z4]=der2spl3(X,Y,1);
[z5,z6]=der2spl3(X,Y,(Y1N-z2)/(z4-z2));
output=z5;
end
```

5-/ Sous-programme VALSPL3.M

```
function [VY,VY1]=valspl3(X,Y,Y2,VX)
*****************************************************************
* détermination de la valeur du spline cubique VY et sa       *
* premiere dérivée VY1 définies par les abcisses des noeuds   *
* X et les coefficients A,B,C et D. Y2 est la valeur du *
* deuxième dérivée du spline.                                  *
*****************************************************************
N=length(X);
IL=1;IH=N;
                while IH>IL+1

  I=fix((IL+IH)/2);
      if X(I)>VX
          IH=I;
      else
          IL=I;
      end
       if IL==IH
            if IL==N
               IL=IL-1;
            else
               IH=IH+1;
```

```
        end
      end
        H=X(IH)-X(IL);
        DX=VX-X(IL);
        A=(Y2(IH)-Y2(IL))/H/6;
        B=Y2(IL)/2;
        C=(Y(IH)-Y(IL))/H-(A*H+B)*H;
        VY=((A*DX+B)*DX+C)*DX+Y(IL);
        VY1=(3*A*DX+2*B)*DX+C;
            end
```

6-/ Sous-programme OPTSPL3.M

```
function output=optspl3(X,Y);

  ************************************************************
  * programme de détermination des coefficients An, Bn,Cn  *
  * et Dn du spline cubique interpolant les données de la  *
  * fonction yi=y(xi) avec optimisation du spline de facon *
  * a minimiser la norme de sa troisième dérivée dans      *
  * l'espace L2 definie par :                              *
  *                        1          (3)    2            *
  *              J = ————————— ||  Y (x)  ||               *
  *                       72               L2             *
  ************************************************************

N=length(X);
**** détermination des termes Delta(n) et S(n) ****

    for k=1:N-1
   DT(k)=X(k+1)-X(k);
   S(k)=(Y(k+1)-Y(k))/DT(k);
   D(k)=Y(k);                          % coefficient Dn
    end

détermination des paramètres Alfa(n);Beta(n) et Gamma(n)
notée par Alfa = Af ; Beta =BT  ; Gamma = GM
**** introduction des conditions initiales ****

Af(1)=0;BT(1)=1;GM(1)=0;Af(2)=0;BT(2)=0;GM(2)=1;
```

```
      for k=3:N-1

   Q=(DT(k)+DT(k-1))/(DT(k-1)+DT(k-2));
   P=Q-2-3*DT(k)/DT(k-1);
   R=S(k)-S(k-1)*(1+Q)+S(k-2)*Q;
   Af(k)=P*Af(k-1)-Q*Af(k-2)+R;
   BT(k)=P*BT(k-1)-Q*BT(k-2);
   GM(k)=P*GM(k-1)-Q*GM(k-2);
     end
% détermination des coéfécients B(1) et C(1) en utilisant les formules décrites dans
% le Chapiitre III-3.

sum11=0;sum12=0;sum22=0;sum1=0;sum2=0;

        for k=1:N-1
  DX3=DT(k).^3;
  sum11=sum11+BT(k).^2/DX3;
  sum12=sum12+BT(k).*GM(k)./DX3;
  sum22=sum22+GM(k).^2/DX3;
  sum1=sum1-Af(k).*BT(k)./DX3;
  sum2=sum2-Af(k).*GM(k)./DX3;
        end
***** détermination de B'(1) et C(1) *****

A(1)=(sum22*sum1-sum12*sum2)/(sum22*sum11-sum12*sum12);
A(2)=(sum1-sum11*A(1))/sum12;

    for k=1:N-1
    A(k)=Af(k)+BT(k)*A(1)+GM(k)*A(2);
    end

K=DT(2)/DT(1);
B(1)=(S(2)-S(1)-A(1)*(2+3*K)-A(2))/(1+K);
C(1)=S(1)-B(1)-A(1);

    for k=1:N-2

  C(k+1)=C(k)+2*B(k)+3*A(k);          %coefficient Cn
  B(k+1)=DT(k+1)*(3*A(k)+B(k))/DT(k);
    end
```

```
**** reconstruction des coefficients An et Bn ****

    for k=1:N-1
    A(k)=A(k)/(DT(k)).^2;              %coefficient An
    B(k)=B(k)./DT(k);                  %coefficient Bn
    end
output=[A' B' C' D'];
end
```

7-/ Sous programme COEF.M

```
function output=coef(T)
    ******************************************************
    * determination des coefficients K0,K1 et K2 de la *
    * relation qui lie la concentration a la fréquence *
    ******************************************************

K0=493.6+(1.66-.014*T)*T;
K1=2.24-(.064-.00095*T)*T;
K2=.014-(5e-5+2.4e-5*T)*T;
output=[K0;K1;K2];
end
```

8-/ Sous programme CTFSIM.M

```
function [F,T]=ctfsim(C,ErrC,ErrF,Tmin,Tmax,DevT,NT)

    ************************************************
    * détermination des valeurs de la fréquence et de la *
    * température pour C donnée                    *
    ************************************************

DeltaT=(Tmax-Tmin)/(NT-1);
T0=Tmin;
rand('uniform')
br=2*rand(1,NT)-1;
C=C+br(1)*ErrC;

        for IT=1:NT
```

```
  T(IT)=T0+br(IT)*DevT;
  z=coef(T(IT));
  F(IT)=z(1)+(z(2)+z(3)*C)*C+br(IT)*ErrF;
  T0=T0+DeltaT;
       end
end
```

9-/ PROGRAMME D'EVALUATION DU SPLINE CUBIQUE POUR LE CALIBRATION DU SYSTEME DE MESURE

9-A-/ PROGRAMME EINTSPL3.M

```
  ********************************************************
  * programme d'évaluation du spline parabolique pour  *
  * le calibration du système de mesure pour l'analyse *
  * ultrasonore des solutions définie par les données: *
  *   (Ci,(Tij,Fij)) pour i=1,...,NC; j=1,...,NT       *
  * et la relation :                                   *
  *              F=K0+(K1+K2*C)*C                      *
  * ou       K0=493.6+(1.66-.014*T)*T                  *
  *          K1=2.24-(.064-.00095*T)*T                 *
  *          K2=.014-(5e-5+2.4e-5*T)*T                 *
  *  C :concentration a mesurer  (%)                   *
  *  F :fréquence de résonance  (Khz)                  *
  *  T :température de la solution (°C)                *
  ********************************************************

Cmax=input('Cmax   : ');
Cmin=input('Cmin   : ');
Tmax=input('Tmax   : ');
Tmin=input('Tmin   : ');
ErrC=input('ErrC   : ');
DevT=input('DevT   : ');
NC  =input('NC     : ');
NT  =input('NT     : ');
NExp=input('Nbre experimentation NExp  :');

ErrC_Int_max_max=0;     : erreur maximale
ErrC_Int_2_av=0;        : erreur dans L2
```

```
            for IExp=1:NExp
DeltaC=(Cmax-Cmin)/(NC-1);
                        for IC=1:NC

      C(IC)=Cmin+(IC-1)*DeltaC;
      [F1,T1]=ctfsim(C(IC),ErrC,0,Tmin,Tmax,DevT,NT);
      F(IC,:)=F1(1,:);
      T(IC,:)=T1(1,:);
      z1=spl3_y2(T(IC,:),F(IC,:));
      F2(IC,:)=z1;
                        end

ErrC_Int_max=ErrC_Int_max_max;
NNC=(NC-1)*4; NNT=(NT-1)*4;
DeltaC=(Cmax-Cmin)/NNC;
DeltaT=(Tmax-Tmin)/NNT;
ErrC_Int_2=0;
            for IC=0:NNC

Cref=Cmin+IC*DeltaC;
                for IT=0:NNT

Tmes=Tmin+IT*DeltaT;
z2=coef(Tmes);
Fmes=z2(1)+(z2(2)+z2(3)*Cref)*Cref;

                        for JC=1:NC
     [F3,R]=valspl3(T(JC,:),F(JC,:),F2(JC,:),Tmes);
     Fest(JC)=F3;
                        end
     z3=spl3_y2(Fest,C);
     C2=z3;
     [Cest,R2]=valspl3(Fest,C,C2,Fmes);
     ErrC_Int=abs(Cest-Cref);

          if ErrC_Int>ErrC_Int_max
                ErrC_Int_max=ErrC_Int;
                Cerr=Cref;
                Terr=Tmes;
           end
```

```
        ErrC_Int_2=ErrC_Int_2+(ErrC_Int)^2;

                end    %{boucle IT}
            end        %{boucle IC}

  ErrC_Int_2=sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));
  ErrC_Int_2_av=ErrC_Int_2_av+ErrC_Int_2;

          end          %{boucle IExp}

ErrC_Int_max_max=ErrC_Int_max
ErrC_Int_2_av=ErrC_Int_2_av/NExp
     end               %{boucle NT}
  end                  %{boucle NC}
```

9-B-/ PROGRAMME EOPTSPL3.M

```
Cmax=input('Cmax   : ');
Cmin=input('Cmin   : ');
Tmax=input('Tmax   : ');
Tmin=input('Tmin   : ');
ErrC=input('ErrC   : ');
DevT=input('DevT   : ');
NC  =input('NC     : ');
NT  =input('NT     : ');
NExp=input('Nbre experimentation NExp  :');

ErrC_Int_max_max=0;
ErrC_Int_2_av=0;
              for IExp=1:NExp
DeltaC=(Cmax-Cmin)/(NC-1);
                    for IC=1:NC
  C(IC)=Cmin+(IC-1)*DeltaC;
  [F1,T1]=ctfsim(C(IC),ErrC,0,Tmin,Tmax,DevT,NT);
  F(IC,:)=F1(1,:);
  T(IC,:)=T1(1,:);
  z1=optspl3(T(IC,:),F(IC,:));
  AF(IC,:)=z1(:,1)';BF(IC,:)=z1(:,2)';CF(IC,:)=z1(:,3)';
  DF(IC,:)=z1(:,4)';
                    end
```

```
  ErrC_Int_max=ErrC_Int_max_max;
  NNC=(NC-1)*4; NNT=(NT-1)*4;
  DeltaC=(Cmax-Cmin)/NNC;
  DeltaT=(Tmax-Tmin)/NNT;
  ErrC_Int_2=0;
              for IC=0:NNC
  Cref=Cmin+IC*DeltaC;
                    for IT=0:NNT
  Tmes=Tmin+IT*DeltaT;
  z2=coef(Tmes);
  Fmes=z2(1)+(z2(2)+z2(3)*Cref)*Cref;

                          for JC=1:NC
[F2,R]=valspl3a(T(JC,:),AF(JC,:),BF(JC,:),CF(JC,:),DF(JC,:),Tmes);
Fest(JC)=F2;
                          end
    z3=optspl3(Fest,C);
    AC=z3(:,1)';BC=z3(:,2)';CC=z3(:,3)';DC=z3(:,4)';
    [Cest,R2]=valspl3a(Fest,AC,BC,CC,DC,Fmes);
    ErrC_Int=abs(Cest-Cref);

              if ErrC_Int>ErrC_Int_max
                 ErrC_Int_max=ErrC_Int;
                 Cerr=Cref;
                 Terr=Tmes;
              end
         ErrC_Int_2=ErrC_Int_2+(ErrC_Int)^2;
                  end   %{boucle IT}
            end         %{boucle IC}

ErrC_Int_2=sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));
ErrC_Int_2_av=ErrC_Int_2_av+ErrC_Int_2;
          end           %{boucle IExp}
ErrC_Int_max_max=ErrC_Int_max
ErrC_Int_2_av=ErrC_Int_2_av/NExp
     end                %{boucle NT}
  end                   %{boucle NC}
```

SPLINE PARABOLIQUE

La fonction d'interpolation du spline parabolique est définie par :

$$Yn(x) = An\ (\ x\text{-}Xn\ )^2 + Bn\ (\ x\text{-}Xn\ ) + Cn$$

ou An,Bn et Cn sont les coefficients du spline qui interpole la séquence yi=f(xi) pour i=1,...,N.

Les sous_programmes utilisés pour l'évaluation du spline parabolique sont les suivants:

*Intspl2 : Il détermine les coefficients An, Bn et Cn pour chaque sub-intervalle

*Valspl2 : Il détermine la valeur du spline pour X donnée ainsi que la valeur de sa première dérivée au meme point.

*Optspl2 : Il détermine les coefficients du spline An, Bn et Cn en minimisant la norme de sa troisème derivée dans l'espace L2.

*Eintspl2 : Le programme d'évaluation du spline parabolique INTSPL2 pour la calibration du système de mesure pour l'analyse ultrasonore des solutions.

*Eoptspl2 : Le programme d'évaluation du spline parabolique OPTSPL2 pour le calibration du système de mesure.

10-/ Sous-programme INTSPL2.M

```
function output=intspl2(X,Y)
  ************************************************************
  * fonction d'interpolation utilisant le  " spline       *
  * parabolique " de la forme :                            *
  *        Yn(x) = An (x-Xn)^2 + Bn (x-Xn) + Cn            *
  * cette fonction est définie pour chaque sub-intervalle  *
  * de la fonction globale  yi=f(xi)  i=1,...,N            *
  ************************************************************
  ************************************************************
  * programme de détermination des coefficients An,Bn et Cn *
  * du spline parabolique pour chaque noeuds (X(n),Y(n)) ou *
  * n=1,...,N en utilisant comme condition initiale la     *
  * connaissance de la première dérivée (dy/dx) pour x=X(1) *
  * notee Y11                                              *
  ************************************************************
N=length(X);
% détermination de la valeur de (dy/dx) pour x=X1 en utilisant la formule % %
% d'interpolation de lagrange pour (X1,Y1),(X2,Y2);(X3,Y3)et (X4,Y4).

**** détermination de Y11=(dy/dx) pour x=X(1) ****

dX1=X(2)-X(1);dX2=X(3)-X(1);dX3=X(4)-X(1);
K1=-1/dX1-1/dX2-1/dX3;
K2=dX2*dX3/dX1/(X(3)-X(2))/(X(4)-X(2));
K3=dX1*dX3/dX2/(X(2)-X(3))/(X(4)-X(3));
K4=dX1*dX2/dX3/(X(2)-X(4))/(X(3)-X(4));
Y11=K1*Y(1)+K2*Y(2)+K3*Y(3)+K4*Y(4);

% détermination des coefficients An,Bn et Cn du spline parabolique qui interpole
% les series de points (X(n);Y(n)) avec n=1,....,N selon la procédure définie dans
% le Chapitre III-3.

***** calcul de Sn et de Delta(Xn) *****
B(1)=Y11;
      for k=1:N-1

    DT(k)=X(k+1)-X(k);
    S(k)=(Y(k+1)-Y(k))/DT(k);
```

```
    A(k)=(S(k)-B(k))/DT(k);                % le coefficient  An
    C(k)=Y(k);                             % le coefficient  Cn
      if k<(N-1)
        B(k+1)=2*S(k)-B(k);                % le coefficient  Bn
      end
       end
output=[A'B'C'];
```

11-/ Sous-programme OPTSPL2.M

```
function output=optspl2(X,Y);
 ************************************************************
 * programme de détermination des coefficients An, Bn,Cn  *
 * du spline parabolique interpolant les données de la    *
 * fonction yi=y(xi) avec optimisation du spline. Dans ce *
 * cas B(1) sera determinée de facon a minimiser la norme *
 * de la seconde derivée du spline dans l'espace L2       *
 * definie par :                                          *
 *                    1                    2              *
 *            J = ------- || Y(2) (x) ||                  *
 *                    8                    L2             *
 ************************************************************
N=length(X);

 **** détermination des termes Delta(n) ,S(n) et Beta(n) ****

sum1=0;sum2=0;

      for k=1:N-1
    DT(k)=X(k+1)-X(k);
    S(k)=(Y(k+1)-Y(k))/DT(k);
    C(k)=Y(k);                             % coefficient Cn
      if  k==1
          Beta=0;
      else
    Beta=Beta+cos(pi*(k-1))*(S(k)-S(k-1));
      end
    sum1=sum1+1/DT(k);
    sum2=sum2+Beta/DT(k);
      end
```

```
**** détermination des coefficients An et Bn ****
A(1)=-sum2/sum1;
B(1)=S(1)-A(1);

      for k=2:N-1

   B(k)=B(k-1)+2*A(k-1);              %coefficient Bn
   A(k)=S(k)-B(k);
   A(k-1)=A(k-1)/DT(k-1);             %coefficient An
     end

A(N-1)=A(N-1)/DT(N-1);
output=[A' B' C'];
end
```

12-/ Sous-programme VALSPL2.M

```
function [VY,VY1]=valspl2(X,A,B,C,VX)
 ************************************************************
 * détermination de la valeur du spline parabolique VY et *
 * sa première dérivée VY1 définies par les abcisses des  *
 * noeuds X et les coefficients A,B et C                  *
 ************************************************************
N=length(X);
IL=1;IH=N;
              while IH>IL+1
 I=fix((IL+IH)/2);
    if X(I)>VX
       IH=I;
    else
       IL=I;
    end
     if IL==IH
        if IL==N
           IL=IL-1;
        else
           IH=IH+1;
        end
      end
        H=X(IH)-X(IL);
```

```
DX=VX-X(IL);
VY=(A(IL)*DX+B(IL))*DX+C(IL);
VY1=2*A(IL)*DX+B(IL);
      end
```

13-/ PROGRAMME D'EVALUATION DU SPLINE PARABOLIQUE POUR LA CALIBRATION DU SYSTEME DE MESURE

13-A/ PROGRAMME EINTSPL2.M

```
*********************************************************
* programme d'évaluation du spline parabolique pour  *
* le calibration du système de mesure pour l'analyse *
* ultrasonore des solutions définie par les données: *
*   (Ci,(Tij,Fij)) pour i=1,...,NC; j=1,...,NT       *
* et la relation :                                    *
*              F=K0+(Ki -K2*C)*C                      *
* ou       K0=493.6+(1.66-.014*T)*T                   *
*          K1=2.24-(.064-.00095*T)*T                  *
*          K2=.014-(5e-5+2.4e-5*T)*T                  *
*  C :concentration a mesurer  (%)                    *
*  F :fréquence de résonnance  (Khz)                  *
*  T :température de la solution (°C)                 *
*********************************************************
Cmax=input('Cmax   : ');
Cmin=input('Cmin   : ');
Tmax=input('Tmax   : ');
Tmin=input('Tmin   : ');
ErrC=input('ErrC   : ');
DevT=input('DevT   : ');
NC=input('NC    :');
NT=input('NT    :');
NExp=input('Nbre experimentation NExp  :');
ErrC_Int_max_max=0;
ErrC_Int_2_av=0;

        for IExp=1:NExp
DeltaC=(Cmax-Cmin)/(NC-1);
```

```
                        for IC=1:NC
  C(IC)=Cmin+(IC-1)*DeltaC;
  [F1,T1]=ctfsim(C(IC),ErrC,0,Tmin,Tmax,DevT,NT);
  F(IC,:)=F1(1,:);
  T(IC,:)=T1(1,:);
  z1=intspl2(T(IC,:),F(IC,:));
  AF(IC,:)=z1(:,1)';BF(IC,:)=z1(:,2)';CF(IC,:)=z1(:,3)';
                        end

ErrC_Int_max=ErrC_Int_max_max;
NNC=(NC-1)*4; NNT=(NT-1)*4;
DeltaC=(Cmax-Cmin)/NNC;
DeltaT=(Tmax-Tmin)/NNT;
ErrC_Int_2=0;
          for IC=0:NNC
Cref=Cmin+IC*DeltaC;
                for IT=0:NNT
Tmes=Tmin+IT*DeltaT;
z2=coef(Tmes);
Fmes=z2(1)+(z2(2)+z2(3)*Cref)*Cref;

                        for JC=1:NC
  [F2,R]=valspl2(T(JC,:),AF(JC,:),BF(JC,:),CF(JC,:),Tmes);
  Fest(JC)=F2;
                        end
    z3=intspl2(Fest,C);
    AC=z3(:,1)';BC=z3(:,2)';CC=z3(:,3)';
    [Cest,R2]=valspl2(Fest,AC,BC,CC,Fmes);
    ErrC_Int=abs(Cest-Cref);

          if ErrC_Int>ErrC_Int_max
              ErrC_Int_max=ErrC_Int;
              Cerr=Cref;
              Terr=Tmes;
          end


   ErrC_Int_2=ErrC_Int_2+(ErrC_Int)^2;

                end   %{boucle IT}
```

```
            end          %{boucle IC}

  ErrC_Int_2=sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));
  ErrC_Int_2_av=ErrC_Int_2_av+ErrC_Int_2;

          end            %{boucle IExp}

ErrC_Int_max_max=ErrC_Int_max
ErrC_Int_2_a=ErrC_Int_2_av/NExp

    end                  %{boucle NT}
  end                    %{boucle NC}
```

13-B/ PROGRAMME EOPTSPL2.M

```
Cmax=input('Cmax   : ');
Cmin=input('Cmin   : ');
Tmax=input('Tmax   : ');
Tmin=input('Tmin   : ');
ErrC=input('ErrC   : ');
DevT=input('DevT   : ');
NC  =input('NC     : ');
NT  =input('NT     : ');
NExp=input('Nbre experimentation NExp  :');

ErrC_Int_max_max=0;
ErrC_Int_2_av=0;
              for IExp=1:NExp

DeltaC=(Cmax-Cmin)/(NC-1);
                        for IC=1:NC
    C(IC)=Cmin+(IC-1)*DeltaC;
    [F1,T1]=ctfsim(C(IC),ErrC,0,Tmin,Tmax,DevT,NT);
    F(IC,:)=F1(1,:);
    T(IC,:)=T1(1,:);
    z1=optspl2(T(IC,:),F(IC,:));
    AF(IC,:)=z1(:,1)';BF(IC,:)=z1(:,2)';CF(IC,:)=z1(:,3)';
                        end

ErrC_Int_max=ErrC_Int_max_max;
```

```
NNC=(NC-1)*4; NNT=(NT-1)*4;
DeltaC=(Cmax-Cmin)/NNC;
DeltaT=(Tmax-Tmin)/NNT;
ErrC_Int_2=0;


            for IC=0:NNC
Cref=Cmin+IC*DeltaC;
                  for IT=0:NNT
Tmes=Tmin+IT*DeltaT;
z2=coef(Tmes);
Fmes=z2(1)+(z2(2)+z2(3)*Cref)*Cref;

                        for JC=1:NC
   [F2,R]=valspl2(T(JC,:),AF(JC,:),BF(JC,:),CF(JC,:),Tmes);
   Fest(JC)=F2;
                        end
  z3=optspl2(Fest,C);
  AC=z3(:,1)';BC=z3(:,2)';CC=z3(:,3)';
  [Cest,R2]=valspl2(Fest,AC,BC,CC,Fmes);
  ErrC_Int=abs(Cest-Cref);

               if ErrC_Int>ErrC_Int_max
                  ErrC_Int_max=ErrC_Int;
                  Cerr=Cref;
                  Terr=Tmes;
               end

  ErrC_Int_2=ErrC_Int_2+(ErrC_Int)^2;

                  end   %{boucle IT}
            end         %{boucle IC}

  ErrC_Int_2=sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));
  ErrC_Int_2_av=ErrC_Int_2_av+ErrC_Int_2;

         end            %{boucle IExp}
ErrC_Int_max_max=ErrC_Int_max;
ErrC_Int_2_av=ErrC_Int_2_av/NExp;
```

PROGRAMMES

```
/*   FUNCTION CTF(T,C)   */

float CTF(Tm,Cr)
float T,C;
{
 float K0,K1,K2,Fm;
 K0=493.6 + (1.66 - .014*T)*T;
 K1=2.24 - (.064 - .00095*T)*T;
 K2=.014 - (.00005 + .000024*T)*T;
 Fm= K0 + ( K1 + K2*C)*C;
 return Fm;
}

/*   PROCEDURE CTFsim(C,Tmin,Tmax,NT,F,T)   */

void CTFsim(C,Tmin,Tmax,NT,F,T)
int NT;
float Tmin,Tmax,C;
float T[],F[];
{
 int IT;
 float DeltaT;
 DeltaT=(Tmax-Tmin)/(NT-1);
  for (IT=1;IT<=NT;IT++)
   {
    T[IT]=Tmin + (IT-1)*DeltaT;
    F[IT]=CTF(T[IT],C);
   }
}

/*   PROCEDURE Der1(x1,x2,x3,x4,y1,y2,y3,y4)   */

float Der1(x1,x2,x3,x4,y1,y2,y3,y4)
float x1,x2,x3,x4,y1,y2,y3,y4;
{
 float c0,c1,c2,c3,dx1,dx2,dx3;
 dx1=x2 - x1;
 dx2=x3 - x1;
 dx3=x4 - x1;
 c0 = -1/dx1 - 1/dx2 -1/dx3;
 c1 = dx2*dx3/dx1/(x3 - x2)/(x4 - x2);
 c2 = dx1*dx3/dx2/(x2 - x3)/(x4 - x3);
```

```
 c3 = dx1*dx2/dx3/(x2 - x4)/(x3 - x4);
 return (c0*y1 + c1*y2 + c2*y3 + c3*y4);
}

/*  PROCEDURE IntSpl2(N,x,y,A,B,C)  */

void IntSpl2(N,x,y,A,B,C)

int N;
float x[10],y[10];
float A[10],B[10],C[10];
{
 int k;
 float S[10],DT[10];
 B[1]=Der1(x[1],x[2],x[3],x[4],y[1],y[2],y[3],y[4]);

   for (k=1;k<=N-1;k++)
   {
    DT[k] = x[k+1] - x[k];
    S[k] = (y[k+1] - y[k])/DT[k];
    A[k] = (S[k] - B[k])/DT[k];
    C[k] = y[k];
       if (k<N-1)
       B[k+1]= 2*S[k] - B[k];
   }
}

/*  FUNCTION ValSpl2(N,x,A,B,C,Vx)  */

float ValSpl2(N,x,A,B,C,Vx)
int N;
float Vx,x[10],A[10],B[10],C[10];
{
int i,il,ih,i1;
float dx;
il=1;
ih=N;
 while (ih>il+1)
 {
   i1=(ih+il)%2;
   i=(ih+il-i1)/2;
    if (x[i]>Vx)
        ih=i;
    else
        il=i;
 }
```

```
    if (il==ih)
     {
      if (il==N)
        il = il-1;
      else
        ih = ih+1;
     }
dx= Vx - x[il];
return (A[il]*dx + B[il])*dx + C[il]) ;
}

/*  PROCEDURE OptIntSpl2(N,x,y,A,B,C)  */

void OptIntSpl2(N,x,y,A,B,C)
int N;
float x[10],y[10];
float A[10],B[10],C[10];
{
 int i,j,k;
 float sum1,sum2,Beta;
 float S[10],DT[10];
 sum1=0.0;
 sum2=0.0;
   for (k=1;k<=N-1;k++)
   {
    DT[k] = x[k+1] - x[k];
    S[k] = (y[k+1] - y[k])/DT[k];
    C[k] = Y[k];
    i=k%2;
       if (k==1)
          Beta=0;
       else
        {
         if (i==1)
         Beta=Beta + (S[k] - S[k-1]);
         else
         Beta=Beta - (S[k] - S[k-1]);
        }
     sum1=sum1 + 1/DT[k];
     sum2=sum2 + Beta/DT[k];
   }
   A[1]=-sum2/sum1;
   B[1]=S[1] - A[1];
       for (j=2;j<=N-1;j++)
       {
       B[j]=B[j-1] + 2*A[j-1];
```

```
        A[j]=S[j]-B[j];
        A[j-1]=A[j-1]/DT[j-1];
        }
        A[N-1]=A[N-1]/DT[N-1];
}

/*   PROCEDURE IntSpl3(N,x,y,A,B,C,D)    */

void IntSpl3(N,x,y,A,B,C,D)
int N;
float x[10],y[10];
float A[10],B[10],C[10],D[10];
{
 int i,j,k;
 float Y11,Y1N;
 float BT[10],GM[10],DT[10],S[10];
 float K,P,R,C2;

/* Computing boundary values of the first derivative of the spline */

 Y11=Der1(x[1],x[2],x[3],x[4],y[1],y[2],y[3],y[4]);
 Y1N=Der1(x[N],x[N-1],x[N-2],x[N-3],y[N],y[N-1],y[N-2],y[N-3]);

   for (k=1;k<=N-1;k++)
   {
    DT[k] = x[k+1] - x[k];
    S[k] = (y[k+1] - y[k])/DT[k];
    D[k] = y[k];                    /* Coefficient Dn */
   }
  BT[1]=0.0;
  GM[1]=Y11;
  BT[2]=1.0;
  GM[2]=0.0;
   for (j=3;j<=N;j++)
   {
    K=DT[j-1]/DT[j-2];
    P=-2.0*(1+K);
    R=3.0*(S[j-1] + K*S[j-2]);
    BT[j] = P*BT[j-1] - K*BT[j-2];
    GM[j] = P*GM[j-1] - K*GM[j-2] + R;
   }

/* Determination of the coefficients An , Bn and Cn  */

  C2 = (Y1N - GM[N])/BT[N];
  C[1]=Y11;
```

```
    for (i=1;i<=N-1;i++)
    {
     C[i+1] = BT[i+1]*C2 + GM[i+1];
     B[i] = ( 3.0*S[i] - C[i+1] - 2.0*C[i])/DT[i];
     A[i] = ( C[i+1] + C[i] - 2.0*S[i])/DT[i]/DT[i];
}
```

/* FUNCTION **ValSpl3**(N,x,A,B,C,D,Vx) */

```
float ValSpl3(N,x,A,B,C,D,Vx)
int N;
float Vx,x[10],A[10],B[10],C[10],D[10];
{
int i,il,ih,i1;
float dx;
il=1;
ih=N;
 while (ih>il+1)
 {
   i1=(ih+il)%2;
   i=(ih+il-i1)/2;
    if (x[i]>Vx)
        ih=i;
    else
         il=i;
  }
    if (il==ih)
      if (il==N)
        il = il-1;
      else
        ih = ih+1;
dx= Vx - x[il];
return (((A[il]*dx + B[il])*dx + C[il])*dx + D[il]);
}
```

/* PROCEDURE **OptIntSpl3**(N,x,y,A,B,C,D) */

```
void OptIntSpl3(N,x,y,A,B,C,D)
int N;
float x[10],y[10];
float A[10],B[10],C[10],D[10];
{
 int i,j,k,h,z;
 float BT[10],GM[10],DT[10],S[10],Af[10];
 float Q,P,R,dX3,K;
 float sum11,sum12,sum22,sum1,sum2;
```

```
 for (k=1;k<=N-1;k++)
 {
  DT[k] = x[k+1] - x[k];
  S[k] = (y[k+1] - y[k])/DT[k];
  D[k] = y[k];                          /* Coefficient Dn */
 }

Af[1]=0.0;
BT[1]=1.0;
GM[1]=0.0;
Af[2]=0.0;
BT[2]=0.0;
GM[2]=1.0;

 for (j=3;j<=N-1;j++)
 {
  Q=(DT[j]+DT[j-1])/(DT[j-1]+DT[j-2]);
  P=Q - 2.0 - 3.0*DT[j]/DT[j-1];
  R=S[j] - S[j-1]*(1+Q) + Q*S[j-2];
  BT[j] = P*BT[j-1] - Q*BT[j-2];
  Af[j] = P*Af[j-1] - Q*Af[j-2] + R;
  GM[j] = P*GM[j-1] - Q*GM[j-2];
 }

/* Determination of the coefficients An , Bn and Cn */

sum11=0.0;
sum12=0.0;
sum22=0.0;
sum1=0.0;
sum2=0.0;
   for (h=1;h<=N-1;h++)
   {
     dX3=DT[h]*DT[h]*DT[h];
     sum11=sum11 + BT[h]*BT[h]/dX3;
     sum12=sum12 + BT[h]*GM[h]/dX3;
     sum22=sum22 + GM[h]*GM[h]/dX3;
     sum1=sum1 - Af[h]*BT[h]/dX3;
     sum2=sum2 - Af[h]*GM[h]/dX3;
   }

A[1]=(sum22*sum1 - sum12*sum2)/(sum22*sum11 - sum12*sum12);
A[2]=(sum1 - sum11*A[1])/sum12;
K=DT[2]/DT[1];
B[1]=(S[2]-S[1]-A[1]*(2+3*K)-A[2])/(1+K);
C[1]=S[1]-B[1]-A[1];
```

```
    for (i=1;i<=N-1;i++)
    {
     A[i] = Af[i] + BT[i]*A[1] + GM[i]*A[2];
     C[i+1] = C[i] + 2.0*B[i] + 3.0*A[i];
     B[i+1] = DT[i+1]*(3.0*A[i]+B[i])/DT[i];
     B[i]=B[i]/DT[i];
    }
    for (z=1;z<=N-1;z++)
    A[z]=A[z]/(DT[z]*DT[z]);
}
```

PROGRAMMES FOR CALIBRATION OF A MEASURING SYSTEM FOR ULTRASONIC ANALYSIS OF SOLUTIONS

1) PROGRAMME FOR EVALUATION OF IntSpl2

```
#include <math.h>
main()
{
 float T[10],F[10];
 float A0[10],B0[10],C0[10];
 float A2[10],B2[10],C2[10];
 float A3[10],B3[10],C3[10];
 float A1[10][10],B1[10][10];
 float C1[10][10];
/*
  Fest  : estimated value of the resonant frequency
  Fmes : measured value of the resonant frequency
  Tmes : measured value of the temperature
  Cest  : estimated value of the concentration
  Cref  : reference value of the concentration
  ErrC_Int_max_max : δx
  ErrC_Int_2_av      : ‖ δx ‖ L2
*/
 float C[10],Fest[10];
 float Cmax,Cmin,dC,DeltaC,Cref,Cest;
 float Tmax,Tmin,DeltaT,Tmes,Fmes;
 double ErrC_Int_max_max;
 double ErrC_Int_2_av;
 double ErrC_Int_max;
 double ErrC_Int_2;
 double ErrC_Int;
 int i,j,k,NC,NT;
 int IC,IT,JC;
 int NNC,NNT;
 Cmax=25.0;
 Tmax=Cmax;
 Cmin=0.0;
 Tmin=Cmin;
 ErrC_Int_max_max=0.0;
 ErrC_Int_2_av=0.0;
 NC=5;
 NT=NC;
 dC=(Cmax - Cmin)/(NC-1);
```

```
        for (j=1;j<=NC;j++)
         {
          C[j]=Cmin + (j-1)*dC;
          CTFsim(C[j],Tmin,Tmax,NT,F,T);
          IntSpl3(NT,T,F,A0,B0,C0);
         for (i=1;i<=NC-1;i++)
             {
              A1[i][j]=A0[i];
              B1[i][j]=B0[i];
              C1[i][j]=C0[i];
             }
      } /* end of j loop */

ErrC_Int_max=ErrC_Int_max_max;
NNC=(NC-1)*4;
NNT=(NT-1)*4;
DeltaC=(Cmax - Cmin)/NNC;
DeltaT=(Tmax - Tmin)/NNT;
ErrC_Int_2=0;

    for (IC=0;IC<=NNC;IC++)
    {
     Cref = Cmin + IC*DeltaC;
        for (IT=0;IT<=NNT;IT++)
        {
         Tmes=Tmin + IT*DeltaT;
         Fmes=CTF(Tmes,Cref);

            for (JC=1;JC<=NC;JC++)
            {
                 for (k=1;k<=NT-1;k++)
              {
                    A3[k]=A1[k][JC];
                    B3[k]=B1[k][JC];
                    C3[k]=C1[k][JC];
              }
               Fest[JC]=ValSpl2(NT,T,A3,B3,C3,Tmes);
             }      /*  end of JC loop  */

            IntSpl2(NC,Fest,C,A2,B2,C2);
            Cest=ValSpl2(NC,Fest,A2,B2,C2,Fmes);
          ErrC_Int=abs(Cest - Cref);

              if (ErrC_Int>ErrC_Int_max)
             ErrC_Int_max=ErrC_Int;
```

```
            ErrC_Int_2 = ErrC_Int_2 + ErrC_Int*ErrC_Int;

          } /* end of IT loop */
      }     /* end of IC loop */

 ErrC_Int_max_max = ErrC_Int_max;
 ErrC_Int_2_av =sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));

}
```

2) PROGRAMME FOR EVALUATION OF OptIntSpl2

The programme for evaluation of the procedure OptIntSpl2 is the same as IntSpl2 except we must change IntSpl2 by OptIntSpl2 only.

3) PROGRAMME FOR EVALUATION OF IntSpl3

```
#include <math.h>
main()

{
 float T[10],F[10];
 float A0[10],B0[10],C0[10],D0[10];
 float A2[10],B2[10],C2[10],D2[10];
 float A3[10],B3[10],C3[10],D3[10];
 float A1[10][10],B1[10][10];
 float C1[10][10],D1[10][10];
 float C[10],Fest[10];
 float Cmax,Cmin,dC,DeltaC,Cref,Cest;
 float Tmax,Tmin,DeltaT,Fmes,Tmes;
 double ErrC_Int_max_max;
 double ErrC_Int_2_av;
 double ErrC_Int_max;
 double ErrC_Int_2;
 double ErrC_Int;
 int i,j,k,NC,NT;
 int IC,IT,JC;
 int NNC,NNT;
 Cmax=25.0;
 Tmax=Cmax;
 Cmin=0.0;
 Tmin=Cmin;
```

```
ErrC_Int_max_max=0.0;
ErrC_Int_2_av=0.0;
NC=5;
NT=NC;
dC=(Cmax - Cmin)/(NC-1);
        for (j=1;j<=NC;j++)
     {
        C[j]=Cmin + (j-1)*dC;
        CTFsim(C[j],Tmin,Tmax,NT,F,T);
        IntSpl3(NT,T,F,A0,B0,C0,D0);

        for (i=1;i<=NC-1;i++)
             {
               A1[i][j]=A0[i];
            B1[i][j]=B0[i];
            C1[i][j]=C0[i];
            D1[i][j]=D0[i];
           }
     } /* end of j loop */

ErrC_Int_max=ErrC_Int_max_max;
NNC=(NC-1)*4;
NNT=(NT-1)*4;
DeltaC=(Cmax - Cmin)/NNC;
DeltaT=(Tmax - Tmin)/NNT;
ErrC_Int_2=0;
          for (IC=0;IC<=NNC;IC++)
          {
           Cref = Cmin + IC*DeltaC;

           for (IT=0;IT<=NNT;IT++)
             {
               Tmes=Tmin + IT*DeltaT;
               Fmes=CTF(Tmes,Cref);
             for (JC=1;JC<=NC;JC++)
             {
                 for (k=1;k<=NT-1;k++)
              {
                  A3[k]=A1[k][JC];
                  B3[k]=B1[k][JC];
                  C3[k]=C1[k][JC];
                  D3[k]=D1[k][JC];
              }              /* end of k loop */

          Fest[JC]=ValSpl3(NT,T,A3,B3,C3,D3,Tmes);
             }              /* end of JC loop */
```

```
            IntSpl3(NC,Fest,C,A2,B2,C2,D2);
            Cest=ValSpl3(NC,Fest,A2,B2,C2,D2,Fmes);
            ErrC_Int=abs(Cest - Cref);

                        if (ErrC_Int>ErrC_Int_max)
                            ErrC_Int_max=ErrC_Int;

            ErrC_Int_2 = ErrC_Int_2 + ErrC_Int*ErrC_Int;

        }    /* end of IT loop */

    }        /* end of IC loop */

 ErrC_Int_max_max = ErrC_Int_max;
 ErrC_Int_2_av =sqrt(ErrC_Int_2/(NNC+1)/(NNT+1));

}
```

4) PROGRAMME FOR EVALUATION OF OptIntSpl3

The programme for evaluation of the procedure OptIntSpl3 is the same as IntSpl3 except we must change IntSpl3 by OptIntSpl3 only.