NANCY : UN LANGAGE DE HAUT NIVEAU POUR LE TRAITEMENT NUMERIQUE DE SIGNAL

E. Verreault, N. Lessard, H.T. Huynh

Département de Génie Electrique, Université Laval, Québec, Canada, G1K 7P4

Le traitement numérique de signal est un domaine où l'utilisation massive du temps de calculateur est monnaie courante. Malheureusement, une grande part du temps utilisé par le chercheur se situe au niveau du développement des programmes. Afin de réduire l'effort fourni lors de cette phase de la recherche, la disponibilité d'un langage de programmation spécifique au traitement de signal est souhaitable. Ce langage doit permettre de considérer les traitements de base au même niveau qu'un opérateur mathématique, et les vecteurs ou les blocs de données en tant qu'entité unifiée. De plus, un niveau d'abstraction supplémentaire pourra être atteint; il évitera au chercheur d'avoir à convertir à tout instant une représentation de données en une autre, au cours d'un même traitement.

Le noyau de base de NANCY est implanté sur un système SUN/UNIX. Un effort est particulièrement apporté sur le maintien d'une portabilité totale du langage sur d'autres systèmes. Aussi, le support de représentation des variables est construit: il inclut la notation matricielle, mono ou multi-vectorielle et scalaire par un seul type de représentation. Le type de transfert des données sur fichier permet une évolution des représentations et des informations connexes sans modification des données. Une surprotection quant aux erreurs internes est également réalisée. La mise des messages d'opérations, d'erreur, des noms de commandes et des paramètres de traitement sur fichier, conduit à une traduction aisée du langage et une action interactive dynamique.

NANCY est ainsi conçu selon les standards pré-établis afin de permettre une uniformité et une continuité d'évolution. L'usager peut donc développer d'une manière indépendante ses algorithmes qui sont utilisés par la suite par le noyau de NANCY sans avoir à repasser par la phase de compilation.

Intensive use of computing time with digital signal processing is a common drawback for all researchers in this field. Although this computing time seems excessive, researchers' time resources are mostly spent during the program development. A time reduction will be achieved with the availability of a programming language specific to the digital signal processing needs. That language must be able to consider the basic DSP treatments on the same level of the mathematical operators, and vectors or data blocks as one unified item. Higher abstraction level should be achieved by avoiding the multiple checks and conversions to do on the data.

NANCY is a programming language built in a multi level fashion; the core is already available on SUN/UNIX workstations, but portability between differents computers is a main feature of the language. The data representation being at the center of NANCY's power, matrixes, single or multiple vectors and scalars are handled by the same representation. File transfers are based on an evolving format independent of the data representation and its associated information evolution. Errors are handled by a trace-back method which covers all the development and program execution bugs. System files handle the execution and error messages, the command names and their parameters list, allowing an easy translation and a dynamic interactive environment for the programmer.

With standards being already defined, NANCY will evolve uniformly and continuously. The programmer will be able to develop his algorithms without going in an infinite loop of compiling ritual and more interesting, he will create an algorithm bank, fully compatible with all the already available resources.

1) INTRODUCTION

Les principaux problèmes rencontrés lors de la programmation d'algorithmes de traitement de signal se présentent à la couche de transition entre le niveau conceptuel et celui de l'implantation; ces deux niveaux étant démarqués par la différence entre l'abstraction des mathématiques et la réalisation des fonctions de calcul. Pis encore, la programmation réalisée, la vérification de sa validité nécessite plusieurs ressources externes: i.e. un support graphique universel ou un accès des données à traiter sur fichier standardisé.

Une première solution est de créer une banque de fonctions qui permet de réduire le problème à celui d'un jeu de bloc, mais qui demande tout de même un effort de compatibilité afin d'y interchanger les données sous les représentations exactes que nécessite chaque fonction. Malheureusement cette solution possède comme désavantage d'impliquer un éternel recommencement de compilations et d'écriture des programmes pour modifier ou changer le traitement à réaliser.

2) LE SYSTEME A COMMANDES OU PAR MENUS

La seconde solution consiste à regrouper sous un même ensemble, les ressouces nécessaires et ce rassemblement doit dénoter les propriétés de versatilité et d'aisance d'utilisation. pour en justifier l'existance. Si cette solution est retenue, de multiples contraintes qui étaient inexistentes lors de l'élaboration primaire, apparaissent après une utilisation de plus en plus massive du système de gestion. Les données à manipuler se retrouvant parmi quatre groupes: les scalaires, les vecteurs, les blocs de vecteurs et les matrices et ces groupes ayant à se côtoyer la plupart du temps pour réaliser un ou plusieurs mêmes traitements, il y a un engorgement des ressources fondamentales: les traitements ayant été préconçus pour ne manipuler que quelques uns des groupes de données.

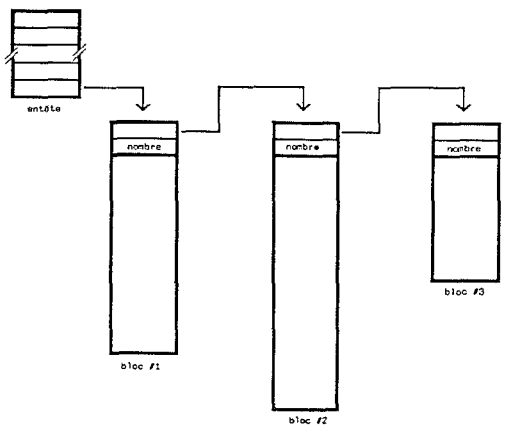
Le système de gestion de ce regroupement de fonction en devient la principale limitation de versatilité; un accès aux fonctions au moyen de commandes ou de menus étant le plus souvent basé sur un support de données par vecteurs de dimension maximale fixe, il est interdicteur, de par sa structure même, de transmission de paramètres d'une commande à l'autre. Si un support de programmation est disponible à l'intérieur du système, il possède malheureusement les mêmes limitations; les éléments de la programmation étant les commandes prédéfinies.

Une évolution majeure de ce type de système peut se réaliser en y ajoutant la notion de variable; à la condition que les variables soient dans la possibilité de supporter les quatre groupes de données ci-haut définies. L'absence de variable étant la différenciation entre le langage à commande et celui d'un langage de programmation, la solution au problème de la programmation des algorithmes de traitement de signal serait donc d'avoir comme seule ressource, un langage de programmation parfaitement adapté.

3) LE LANGAGE DE PROGRAMMATION

Baser un langage de programmation pour le traitement de signal sur des variables n'est de gain autre que celui d'ajouter des fonctions internes à un langage de programmation général. La nécessité de considérer les quatre éléments de base du traitement de signal en tant qu'entité uniforme, appelées objets, est le saut d'abstraction qui pourra permettre de décharger le programmeur de la tâche d'adapter les fonctions à ses besoins; c'est le noyau fonctionnel du langage qui se doit de s'adapter aux besoins de l'utilisateur. La structure des objets du langage de programmation de traitement de signal NANCY a été élaborée afin de permettre une représentativité globale et une possibilté d'adaptation sans restrictivité.

A la racine de chaque objet de NANCY, est une entête qui permet de caractériser les données; elle ne contient aucune information sur le nombre de données, seulement l'information du domaine et unités des données de l'objet; de l'entête on en déduit la localisation du premier vecteur des données. L'information de la localisation du second vecteur, s'il y a, est fourni à partir du premier vecteur et ainsi de suite (FIG 1). La représentation d'un scalaire est réalisée par un vecteur unique de dimension unitaire et un bloc de vecteurs ou une matrice par une combinaison de vecteurs. Lors de la demande d'exécution d'un traitement, seule l'entête est transmise comme paramètre; un premier niveau de fonction accède les données de l'objet et en décide du type de manipulation selon le groupe spécifique de chaque l'objet. Cette fonction par la suite appelle une ou plusieurs fois la fonction réalisant le traitement proprement dit avec le ou les paramètres requis. Ainsi, si un traitement requiert un scalaire et qu'un vecteur y est transmis, alors le traitement s'effectuera avec chacun des éléments du vecteur, correspondant à un ensemble de traitements répétifs à paramètres variables.



structure des objets de NANCY
(FIG 1)

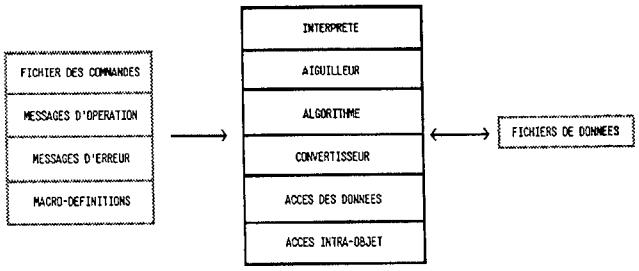
4) LE LANGAGE NANCY

La couche externe de NANCY, soit l'interprète des expressions et commandes a de beaucoup été inspirée par celle du langage de programmation "C": il n'y existe presque pas de fonctions de haut niveau, spécifiques à l'interprète, ce sont dans l'ensemble des fonctions déclarées ou qui pourraient être déclarées au niveau du langage NANCY. La différence entre le langage "C" et NANCY est la même que celle qui sépare le niveau d'abstraction d'une variable et celui d'un objet.

L'information syntaxique et paramétrique des fonctions propres a été localisée dans un fichier appelé fichier des commandes. De ce fichier, l'utilitaire d'aide intéractif en extrait ses ressources et de même que pour les paramètre de chaque fonction: y sont déposées les valeurs extrêmes ainsi que les valeurs typiques permettant de rendre un paramètre facultatif. La localisation de cette information au niveau du fichier des commandes permet de complèter les paramètres non-spécifiés par l'usager en extrayant l'information associée et permet une élimination quasi-totale des vérifications de validité par la fonction interne. L'importance de ce fichier apparaît en tant que centralisateur de la connaissance du domaine des paramètres. Pour chaque paramètre manquant, une demande à l'usager sera faite ou une substitution automatique sera assignée par la seule information contenue dans ce fichier. Deux autres fichiers de système sont présents, celui des messages d'erreurs et d'exécution.

L'évolution du langage est assurée par deux méthodes radicalement différentes: la première se situe au niveau du développement des ressources personnelles par la présence de deux banques, locale et globale, de nouvelles fonctions basées sur la syntaxe de NANCY, la seconde facilite le développement des ressources internes de NANCY. L'information sur les fonctions internes étant définies dans le fichier des commandes, une section a étée réservée pour des commandes non-définies qui feront appel à des programmes externes partageant les mêmes ressources. Une fois les programmes externes entièrement fonctionnels, une simple substitution de la section de transfert de contrôle permettra d'inclure le nouvel algorithme comme une partie intégrale de NANCY.

L'inclusion en est facilitée grâce à la disposition par couche des niveaux fonctionnels, correspondant chacune à un domaine d'abstraction. Ainsi, à la base de NANCY, sont les fonctions d'accès au niveau des pointeurs mémoire des objets; le suivant, l'accès aux données proprement dites des objets. La troisième couche ayant pour tâche de réaliser les conversions entre les diverses représentations en traitement de signal et la couche suivante, celles des algorithmes ou fonctions de traitement. La couche supérieure est l'interprète des commandes ou opérations et est localisée immédiatement au dessus de la couche d'aiguillage des traitements. Cette couche supérieure pourra être supplantée à tout moment; cette particularité rend NANCY spécialement adapté pour une conversion vers un système expert ou une transformation vers un système à menus à orientation didactique.



structure en couches de NANCY
(FIG 2)

5) LA SYNTAXE DES COMMANDES

EXEMPLE

La fonction demandée est de modéliser un ensemble de données par la méthode de covariance en prédiction linéaire. Comme il est bien connu, il faut spécifier l'ordre de modélisation et l'algorithme peut retourner trois informations distinctes: le modèle auto-regressif sous forme de filtre en treillis et transversal et les erreurs résiduelles des prédictions. S'il est désiré de connaître l'ordre optimal de modélisation, l'erreur résiduelle de prédiction est l'information utile avec laquelle on utilise une équation de critère d'ordre; d'autre part, si c'est le modèle qui est requis, la vérification de la stabilité du modèle est plus aisément réalisée avec le filtre en treillis que transversal.

Dans le langage NANCY, l'appel à l'exécution de l'algorithme est de la forme suivante:

```
( ERR:nom1, LAT:nom2, COEF:nom3 ) =
        estimation a-r covariance ( ORD:nom4, DATA:nom5 );
```

A la différence de la plupart des langages de programmation, l'aiguillage pour l'échange des objets est réalisé par l'intermédiare d'étiquettes descriptives. Cette utilisation permet de faire disparaître l'imposition de spécifier tous les paramètres et de plus, assure une bonne lisibilité pour une tierce personne. Ainsi, on peut omettre l'étiquette ORD: et son nom d'objet associé; ce faisant, NANCY assignera une valeur prédéterminée pour modéliser le ou les vecteurs de données de l'objet spécifié par l'étiquette DATA:, soit nom5. De même, si on ne désire que certains résultats spécifiques de cet algorithme, on ne déclare que les étiquettes et les noms d'objet associés qui décrivent l'information.

L'information contenue dans le fichier des commandes permet de transmettre à la fonction d'estimation selon l'algorithme de covariance, tous les paramètres sans excep-

tion. Ainsi, pour les paramètres d'entrée nécessaires et absents de la déclaration, une question sera posée à l'usager et le texte de requête est extrait directement du fichier des commandes

```
#20
estimation a-r covariance
modelisation auto regressive covariance
$DESCR
permet d'estimer le modele auto-regressif d'une sequence de donnees par la
methode de prediction lineaire basee sur la matrice de covariance.
$OUTPUT
ERR:                                                    !erreurs residuelles de prediction
LAT:                                                    !coefficients de filtre en treillis du modele
COEF:                                                   !coefficients de filtre tranversal du modele
$INPUT
DATA:double                                             !donnees a modeliser
ORD:int          MIN:0    MAX:NMB@DATA    TYP:NMB@DATA/3   !ordre de modelisation
```

Extrait du fichier des commandes
(FIG 3)

6) LES COMMANDES INTRINSEQUES

Un langage de programmation serait incomplet s'il ne possèdait pas de structures de contrôle. NANCY supporte intrinsèquement, indicatif de l'absence de référence au fichier des commandes, les trois éléments de contrôle suivant: FOR et les deux couplets WHILE-DO et IF-ELSE. La présence d'éléments de contrôle fait appel à la notion de variable logique; dans le cas de NANCY, les variables logiques sont supportées au niveau de la structure de l'objet, par valeurs numériques référenciées à la valeur nulle. Les six opérations de comparaison mathématique sont disponibles et qui dit comparaison, implique une référence entre deux valeurs numériques.

Etant donné que les objets de NANCY supportent dimensionnellement, au plus un tableau d'entités de dimension de deux (matrice ou bloc de vecteurs), l'accès aux valeurs est réalisé par une méthode indicielle. La présence de nombres complexes et de numérateur et dénominateur n'est pas considérée au niveau indiciel; l'utlisation de fonctions du fichier des commandes, permet d'accèder chacun des items constitutifs.

7) CONCLUSION

L'utilisation d'un langage de programmation spécifique au traitement numérique de signal permettra de centraliser l'ensemble des ressources nécessaires de ce domaine. De plus, un langage tel NANCY permet au chercheur de disposer d'un environnement de travail bien adapté. Une telle approche libère programmeur et chercheur d'adapter le graphisme, l'accès sur fichier et les conversions des réprésentations utilisées en traitement de signal, tout en lui donnant un accès immédiat à une librairie de fonctions en perpétuelle expansion.

□ Brian W. Kernighan, Dennis Ritchie
The C Programming Language
Prentice-Hall Software Series

□ LEX-A Lexical Analyser Generator
Sun Microsystems Documents

□ YACC-Yet Another Compiler-Compiler
Sun Microsystems Documents

□ André Zaccarin
DIES, un Ditacticiel D'Estimation Spectrale
Lab. de Traitement Numérique de Signal
Dept. Génie Electrique, Université LAVAL

□ Marcellin Cusson, Germain Drolet, Richard Lepage
SITS : Système Intéractif pour le Traitement de Signal
Centre de Recherche de Valcartier

□ Eric Verreault
SITRANS: Manuel de L'usager
Lab. de Traitement Numérique de Signal
Dept. Génie Electrique, Université LAVAL