Algebraic Methods for the Analysis and Design of Time-Frequency Signal Processing Algorithms

Domingo Rodríguez
Electrical and Computer Engineering Department
University of Puerto Rico, Mayaguez PR 00681-5000
E-mail: domingo@rmece01.upr.clu.edu

Jaime Seguel
Mathematics Department
University of Puerto Rico, Mayaguez PR 00681-5000
E-mail: jseguel@rummat1.upr.clu.edu

Edgardo Cruz
Electrical and Computer Engineering Department
University of Puerto Rico, Mayaguez PR 00681-5000
E-mail: cruz@rmece01.upr.clu.edu

Abstract - This work presents some results on the study of algebraic methods for the analysis and design of time-frequency signal processing algorithms. A time-frequency signal is defined as a signal whose spectral characteristics vary with time. Time-frequency signal processing is the mathematical treatment of signals with the objective of extracting relevant information. The processing of time-frequency signals is accomplished through the development of suitable algorithms. The work concentrates on the analysis, design, and modification of time-frequency algorithms for machine implementation, and on the development of an environment to assist on this implementation. This environment is termed a computational mathematics environment (CME).

I. INTRODUCTION

Time-frequency signals appear in many scientific and engineering application areas such as speech signal processing, SONAR/RADAR signal processing, seismic signal processing, image processing, biomedical signal processing, Fault diagnostics signal processing, Holographic interferometry signal processing, etc. For many years signal processing tools such as the discrete short-time Fourier transform (DSTFT), the discrete finite ambiguity function (DFAF), and the discrete Wigner-Ville distribution (DWVD) have been used successfully in these areas. New computational tools such as the discrete wavelet transform (DWT) are now reaching the field of signal processing. This work presents algorithms for the DSTFT, DFAF, DWVD, and DWT. The algorithms are implemented in workstations using C-language and numerical computation packages. They are also analyzed for their inherent parallel and vector processing structures for implementation on multiprocessing environments. One of the goals of this work is to obtain low-cost parallel implementation of signal processing algorithms.

The central idea of this work is the analysis and design of time-frequency signal processing algorithms. These algorithms operate on finite discrete signals (one-dimensional or two-dimensional, as the case may be), called input signals, to produce an output or processed signal. An prime objective is to process input signals in a fast and efficient manner. Thus, it is desired to obtain algorithm implementations on available machines which can achieve this objective.

Implementing an algorithm on a specific machine for optimal results is not a trivial matter. Several reasons can be provided to support this claim. One reason is that the performance of an algorithm varies from machine to machine, and it is desirable to identify on which machine it performs best. Another reason is that, for a specific machine, many variants of a given algorithm must be tried in order to reach an optimal implementation.

II. COMPUTATIONAL MATHEMATICS ENVIRONMENT

The concerns stated above suggest the development of a methodology for the machine implementation of time-frequency signal processing algorithms. This work presents a methodology based on algebraic methods. To describe this methodology, several definitions and concepts are now presented.

An algorithm can be defined, in a loose manner, as a formulated computational procedure, with a finite number of steps, which acts on elements of a set called input set to produce an output which belongs to an output set. These formulations can take the form of mathematical expressions as it is the case for this work. All mathematical formulations of algorithms are considered to belong to an environment termed a computational mathematics (CME) environment. The CME consists essentially of an input set, an output set, a set of mathematical operators, and a set of rules for these operators. A CME is defined as a software based environment with special subsets called computational mathematics frameworks (CMF). A CMF is a set formed by a mathematical formulation of an algorithm, presented in what is called its canonical or standard form, and all its possible variants. A mathematical formulation is called a variant of a canonical formulation if this variant can be obtained from the canonical formulation through algebraic manipulations or techniques.

Consider the scenario (see Figure 1) where it is desired to implement an algorithm on a given machine (we shall use the generic term computational hardware structure (CHS) to refer to any arbitrary machine). The computational mathematics environment (CME) can assist in this implementation in the following manner. Information about inherent CHS attributes, such as main memory configuration, size of main memory, number of processing units, types of processing units, size of cache, compiler specifications, programming languages, etc., can be acquired in the form acquired by the CME, some of them in the form of parameters. A computational mathematics framework (CMF) can then be constructed where the algorithm and some of its variants are formulated using the acquired information from the CHS. Each variant is then implemented and tested for optimality based on certain performance criteria. A variant will then be chosen which best satisfies these performance criteria.

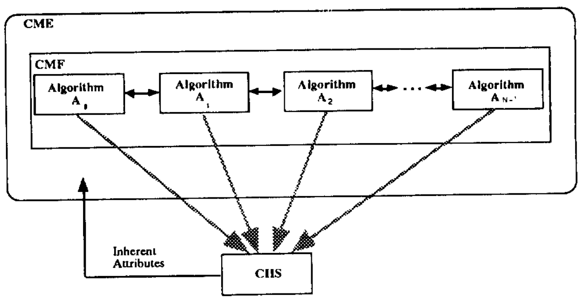

Figure 1

Thus, an algorithm can be referred to as a mathematical formulation describing a composition of a finite number of operators, this composition acting on an input element or input signal to produce a processed signal. To obtain mathematical formulations of a given algorithm, structures, called computational mathematics structures (CMS), are identified on the input and output data sets.

As stated above, this work concentrates on the analysis, design, and modification of time-frequency algorithms for machine implementation, and on the development of an environment to assist on this implementation. This environment is termed a computational mathematics environment (CME). Its main features are the characterization of algorithms and their variants from the algebraic structure point of view.

Consider another scenario where several computational hardware structures (CHS) are provided (see Figure 2) and it is desired to implement a mathematical formulation of an algorithm on these various machines. In this case the CME can acquire the inherent attributes from each of the machines and then determine which of the computational hardware structures effect the most efficient implementation of the given algorithm formulation.

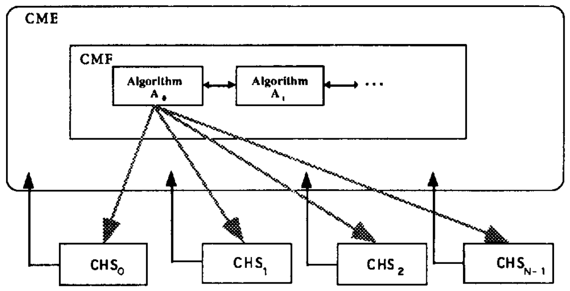

Figure 2

As it was defined above, an algorithm can be referred to as a mathematical formulation describing a composition of a finite number of operators, this composition acting on an input element of input signal to produce a processed or output signal. To obtain mathematical formulations of a given algorithm, structures, called computational mathematics structures (CMS), are identified on the input and output sets.

III. COMPUTATIONAL MATHEMATICS STRUCTURES

As an example of the algebraic structures used in this work, tensor product algebra, convolution operations, Hadamard products, direct sums, permutation operations, etc., are presented. In the case of tensor product algebra, properties of this algebra and the associated permutation matrices are used to study the structure and the computational complexity of algorithms. as well as to formulate variants [2]. Signals to be processed are interpreted as living over finite Abelian groups. This approach results in an algebraic characterization of all the algorithms presented in this work from the point of view of computational complexities and implementation frameworks.

The ideas presented in this section do not intend to exhaust the rich variety of ways in which symbolic processing based on well-known algebraic settings can help in algorithm design. Indeed, in the following paragraphs, we just point out to what we believe are the most natural and in some sense, most elementary examples of mathematical structures underlying the modification and adaptation of algorithms for computing $v = Au$, A a square matrix and u a generic vector, to different machine architectures.

Let $A \otimes B = [a_{kl}B]$, where $A = [a_{kl}]$,

$$\text{Let } J_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and let us denote by

$$P_1 = I_2 \otimes I_2 \otimes J_2,$$
$$P_2 = I_2 \otimes J_2 \otimes I_2,$$
$$P_3 = J_2 \otimes I_2 \otimes I_2.$$

These permutations represent nearest neighbor exchanges in a 3-D hypercube. In fact,

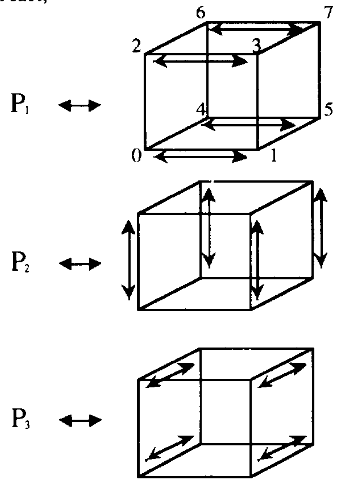

Figure 3

the group generated by P_1, P_2 and P_4 consist of $P_0 = I_2 \otimes I_2 \otimes I_2$, P_1, P_2, P_4 and $P_3 = P_1P_2$, $P_5 = P_1P_4$, $P_6 = P_2P_4$ and $P_7 = P_1P_2P_4$. Any parallel data transmission in a 3-D hypercube correspond to a particular P_j.

Any 8×8 matrix A can be expressed as

$$A = \sum_{j=0}^{7} D_j P_j$$

where D_j is a diagonal matrix. This decomposition provides important insights for the design of algorithms for computing

$$v = Au$$

in a 3-D hypercube. For example, let's consider the 8-point Cooley-Tukey FFT. In matrix terms, the algorithm consist of computing

$$\hat{u} = F_8 u$$

through the matrix factorization $F_8 = S_3S_2S_1B_8$

where B_8 is the 8-point bit-reversal permutation and

$$S_1 = T_1(I_2 \otimes I_2 \otimes F_2),$$
$$S_2 = T_2(I_2 \otimes F_2 \otimes I_2),$$
$$S_3 = T_3(F_2 \otimes I_2 \otimes I_2).$$

where T_j is the diagonal matrix representing the "twiddle" or phase factor [1]. Using decomposition of the type shown above

$$S_1 = T_1\left[(I_2 \otimes I_2 \otimes I_2) + (I_2 \otimes I_2 \otimes J_2)D_1\right]$$

$$= T_1(P_0 + P_1D_1)$$

$$S_2 = T_2\left[(I_2 \otimes I_2 \otimes I_2) + (I_2 \otimes J_2 \otimes I_2)D_2\right]$$

$$= T_2(P_0 + P_2D_2)$$

and

$$S_3 = T_3\left[(I_2 \otimes I_2 \otimes I_2) + (J_2 \otimes I_2 \otimes I_2)D_4\right]$$

$$= T_3(P_0 + P_4D_4)$$

Each of these expressions contains only one nearest neighbor exchange permutation. Thus, each step in the 8-point Cooley-Tukey FFT requires only one parallel exchange in a 3-D hypercube.

The algebraic structure of group with which the set $\{P_0, \ldots, P_7\}$ is endowed allows the expression of each permutation in this set in terms of compositions of nearest neighbor exchanges. This property, in its turn, allows for convenient nestings of nearest neighbor exchanges able to eliminate more complicated data movements. For example, if A is a thick 8×8 matrix, then none of the diagonal D_j is null and therefore we get

$$A = D_0 + P_1D_1 + P_2D_2 + P_3D_3 + \ldots + P_7D_7$$

Now, using the group structure of $\{P_0, \ldots, P_7\}$ we can write

$$A = \left[(D_0 + P_1D_1) + P_2(D_2 + P_1D_3)\right] +$$

$$P_4\left[(D_4 + P_1D_5) + P_2(D_6 + P_1D_7)\right]$$

which again contains only nearest neighbor exchanges.

Since the above formula holds for every matrix A, an important question regarding algorithm design is whether there exist a matrix factorization of A, let's say $A = S_n \cdot \ldots \cdot S_1$, in which each factor A_j decomposes with fewer nearest neighbor exchanges than A, as it happens with F_8 in the Cooley-Tukey FFT. The answer to this question can be provided by a symbolic treatment of A.

In the same way in which the group $\{P_0, \ldots, P_7\}$ characterizes the 3-D hypercube architecture other computational architectures can also be characterized by groups of permutations. For example, 8 processors linked by a ring can be characterized by the group generated by the cyclic-shift permutation

$$C_8 = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ & & & & & \\ 0 & 0 & 0 & 0 & \ldots & 1 \\ 1 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}.$$

This group consists of all the powers of C_8, namely $\{C_8^0, C_8^1, \ldots, C_8^7\}$. As in the case of the 3-D hypercube, any 8×8 matrix A admits a decomposition as:

$$A = \sum_{j=0}^{7} D_j C_8^j$$

and the nesting formula

$$A = D_0 + (D_1 + (D_2 + (D_3 + (D_4 + (D_5 +$$

$$(D_6 + D_7C_8)C_8)C_8)C_8)C_8)C_8$$

allows the expression of all permutations involved in computing

$$v = Au$$

in a ring, in terms of the simplest permutation.

The group generated by the tensor product $C_8 \otimes C_8$, this is the group of permutations of the form:

$$\{C_8^i \otimes C_8^j : 0 \le i, j \le 7\}$$

corresponds to a wrap-around 8×8 mesh.

This algebraic set up can be exploited in many ways. Symbolic processing can provide different factorizations for a given matrix A, and these factors can be tested for proficiency in different architectures, through the above mentioned expressions in terms of groups of permutations.

Different decompositions (d_j maps in Figure 4 below) give rise to different algorithms. Also, a fixed algorithm can be tested symbolically in different architectures through algebraic maps between the group of permutations that characterize the given architectures. These maps are represented by t_j in the following figure. In general, a procedure $v = Au$ will be well adapted to a given architecture if there are permutations P, Q and "well structured" factors $S_1, \ldots, S_n$ such that $v = PBQu$, where $B = S_n \cdot \ldots \cdot S_1$, n is small, and each S_j is expressible in terms of the group permutations associated to the architecture with few basic permutations and full diagonal matrices.

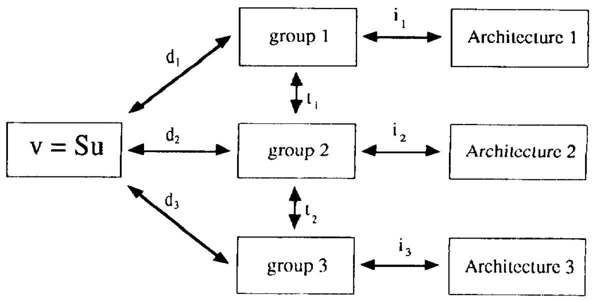

Figure 4

t_j, $j = 1, 2$:	group relation maps
d_j, $j = 1, 2, 3$:	decomposition of A in terms of the elements in group j.
i_j, $j = 1, 2, 3$:	identification maps.

IV. ALGEBRAIC REPRESENTATIONS OF TIME-FREQUENCY FRAMEWORKS

As an example of the type of algorithms presented in this work, the discrete finite ambiguity function [3] is briefly discussed below, and an algorithm and a variant are described for the computation of this function.

Let x be a transmitted signal. Let y be the returned echo after the signal bounces off a stationary object or an object moving at a constant velocity. The discrete finite ambiguity function (DFAF) can then be used to estimate the range and velocity parameters of the moving object. It is defined as the absolute value of the following function

$$A(x,y)[m,k] = \sum_{n=0}^{N-1} x[n]y^*[m+n]e^{-j2\pi\left(\frac{kn}{N}\right)},$$

This function can be computed, for instance, as a filtering operation (hence, the term the filter method) of the incoming signal y with a weighted version of the transmitted signal x. This is accomplished by forming the product $z_n[k] = x[n]e^{-j2\pi\left(\frac{kn}{N}\right)}$. It follows that

$$A(x,y)[m,k] = \sum_{n=0}^{N-1} z_n[k]y^*[m+n]$$

The function can also be computed by forming the Hadamard product $u_n[m] = x[n] \odot y_n^*[m]$ for $y_n[m] = y[m+n]$, and then taking the discrete Fourier transform (DFT) of the result (hence, the term the DFT method). This produces the following expression

$$A(x,y)[m,k] = DFT\{u_n[m]\} = U_k[m]$$

As an example of how to develop a computational mathematics framework (CMF) the filter method is chosen to produce a standard algorithm mathematical formulation and some variants. A functional diagram is presented below (see Figure 5) for an algorithm to compute discrete finite ambiguity function using the filter method [2]. The functional diagram describes the action of a composition of linear, finite dimensional operators on finite complex sequences which can be considered elements of the input set of a computational mathematics environment (CME).

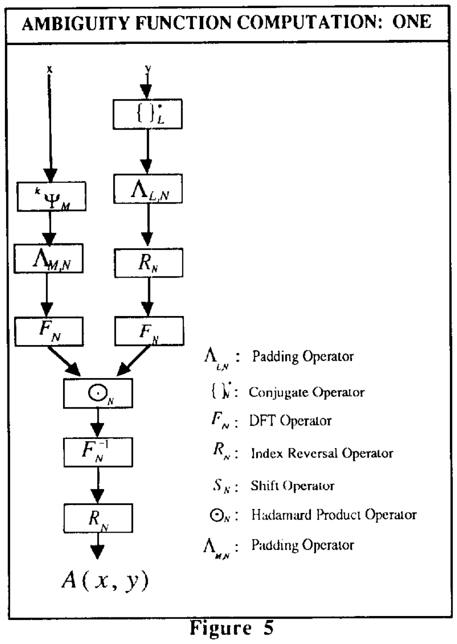
AMBIGUITY FUNCTION COMPUTATION: ONE
x
y
$\{\ \}^*_L$
${}^k\Psi_M$
$\Lambda_{L,N}$
$\Lambda_{M,N}$
R_N
F_N
F_N
$\odot_N$
F_N^{-1}
R_N
$A(x,y)$
$\Lambda_{L,N}$: Padding Operator
$\{\ \}^*$: Conjugate Operator
F_N : DFT Operator
R_N : Index Reversal Operator
S_N : Shift Operator
$\odot_N$: Hadamard Product Operator
$\Lambda_{M,N}$: Padding Operator

Figure 5

In the functional diagram depicted below (see Figure 6), a variant of the algorithm described above for computing the ambiguity function using the filter method is presented. This variant was obtained by using algebraic methods. It was noticed, for instance, that the character operator ${}^k\Psi_N$, the discrete Fourier operator F_N, and the shift operator S_N are related through the following identity

$$F_N\,{}^k\Psi_N = S_N F_N$$

Using properties of linear operators we can produce other variants. For example, the discrete Fourier transform (DFT) operator can be represented as a composition of other linear operators [11]. There are many representations for the DFT operator as was described in the section on computational mathematics structures.

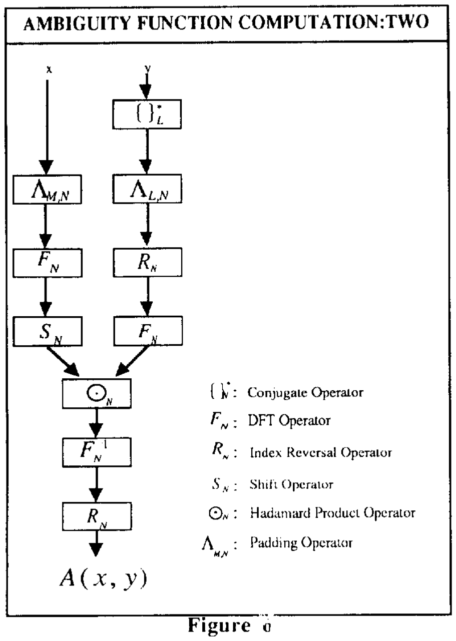
AMBIGUITY FUNCTION COMPUTATION:TWO
x
y
$\{\ \}^*_L$
$\Lambda_{M,N}$
$\Lambda_{L,N}$
F_N
R_N
S_N
F_N
$\odot_N$
F_N^{-1}
R_N
$A(x,y)$
$\{\ \}^*$: Conjugate Operator
F_N : DFT Operator
R_N : Index Reversal Operator
S_N : Shift Operator
$\odot_N$: Hadamard Product Operator
$\Lambda_{M,N}$: Padding Operator

Figure 6

V. CONCLUSION

This has presented some results of algebraic methods used as a tool in the analysis and design of time-frequency algorithms for signal processing. The experience acquired by the authors in this project points in the direction of developing a robust computational mathematics environment as a means to enhance the algorithm design an implementation process. The essential components of this environment were described, and examples where given on how to use this environment in the formulation of fast algorithms for the computation of the discrete Fourier transform and the discrete finite ambiguity function. In the case of the ambiguity function, it was shown how to use simple properties of the algebra of linear operators to obtain a variant of an algorithm from a given mathematical formulation.

A typical researcher in the area of signal processing or applied computational mathematics, for instance, is more interested in using algorithms to solve the engineering and scientific problems rather than becoming an skillful programmer or expert on a particular machine. Tools must then be provided to this researcher so that he can reduce the time spent in analyzing, designing, modifying, and implementing algorithms to satisfy his or her particular needs. The idea of rapid prototyping signal processing systems to address specific problems in engineering and scientific applications is becoming amenable to many researchers. The authors believe that a computational mathematics environment is one of such tools.

A CME was defined above as a software based environment used as a tool in the development of algorithms. The authors have gained some experience in configuring some aspects of CME environments in the numerical computation system MATLAB [7], the visual programming environment LabVIEW [8], and the computer algebra systems MAPLE [9] and MATHEMATICA [10]. None of these systems, however, are adept to implement a CME in a fully integrated manner. MATLAB, for instance, lacks the needed symbolic manipulation capability for it is a numerical computation system. LabVIEW provides visual programming capabilities but also lacks in symbolic manipulation. The computer algebra systems MAPLE and MATHEMATICA provide excellent algebraic manipulation; but they still lack the abstraction capability needed for certain type algebraic analysis as described in [1] and [2]. For instance, analysis of the action of operators on vectors representing elements of the input set of the CME environment has been a problem area.

VI. REFERENCES

[1] J. Seguel, J. Barety "Matrix Algebra and Hypercube Parallel Transmissions." Proc. 13th IMACS, Vol.2, pp 787-788, 1991.

[2] J. Johnson, R. Johnson, D. Rodríguez, R. Tolimieri, "A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms Various Architectures." Journal of Circuits, Systems and Signal Processing, Birkhäuser, Vol. 9, No. 4, 1990.

[3] R. Tolimieri, S. Winograd, "Computing the Ambiguity Surface." IEEE ASSP Vol. 33, pp. 1239-1245, Oct. 1985.

[4] C. Van Loan, "Computational Frameworks for the Fast Fourier Transform." Frontiers in Applied Mathematics, SIAM, Philadelphia, 1992.

[5] W. Kozek, "Time-frequency Signal Processing Based on the Wigner-Weyl Framework." Signal Processing, Vol 29, pp. 77-92, Elsevier, 1992.

[6] M. J. Shensa, "The Discrete Wavelet Transform: Wedding the À Trous and Mallat Algorithms." IEEE Transactions on Signal Processing, Vol. 40, No. 10, Oct. 1992.

[7] MATLAB for SUN Workstations, The Math Works, Inc., Natick, Massachusetts, 1992.

[8] LabVIEW's User Manual, National Instruments Corporation, Austin, Texas, 1991.

[9] Bruce W. Char, et al. "MAPLE V." Springer-Verlag 1992.

[10] Stephen Wolfram, "MATHEMATICA." Addison-Wesley, 1991.

[11] D. Rodríguez, "Tensor Product Algebra as a Tool for VLSI Implementation of the Discrete Fourier Transform." IEEE ICASSP '91, Vol. 2, pp. 1025-1028, Toronto, Canada, 1991.