EVALUATION ALGORITHM FOR A DECOMPOSED SIMPLICIAL PIECEWISE-LINEAR FORMULATION

V.M. Jiménez-Fernández[1], J. Agustín-Rodríguez[2], P. Marcelo-Julián[2], O. Agamennoni[2]

[1]National Institute for Astrophysics, Optics and Electronics, Department of Electronics, Luis Enrique Erro No.1, Santa María Tonantzintla, P.O. 72840 Puebla, México.
[2]Universidad Nacional del Sur, Departamento de Ingeniería Eléctrica y de Computadoras, Av. Alem No. 1253, P.O. 8000 Bahía Blanca, Argentina.

ABSTRACT

In this work an algorithm for the evaluation of N-dimensional piecewise-linear (PWL) functions is presented. The type of PWL representation which is considered is the denominated simplicial representation that is defined in a N-dimensional domain partitioned by hyperplanes and divided into simplices. The algorithm performs a local function computation into the specific simplex where the evaluation point is found. The simplicial PWL (S-PWL) interpolating equations are collected into a matrix system which adopts the form of the decomposed PWL models. The algorithm works directly with this decomposed model and determines the value of the S-PWL function simply by its values on the vertices.

RESUMEN

En este trabajo se presenta un algoritmo para la evaluación de funciones lineales a tramos (LAT) N-dimensionales. El tipo de representación LAT que es considerado es el denominado descripción simplicial que se define en un dominio N-dimensional particionado por hiperplanos y dividido en símplices. El algoritmo realiza un cómputo local de la función en el símplice específico donde se encuentra el punto de evaluación. Las ecuaciones de interpolación de la representación LAT simplicial (LAT-S) son colectadas en un sistema matricial que adopta la forma de los modelos LAT descompuestos. El algoritmo trabaja directamente con este modelo descompuesto y determina el valor de la función LAT-S simplemente mediante su valor en los vértices.

KEY WORDS: Simplicial, Piecewise-Linear, Evaluation

1. INTRODUCTION

Piecewise linear (PWL) functions are widely used in circuit theory [1-3], image processing [4], and system identification [5]. The evaluation of this type of functions has been approached in different ways by diverse algorithms such as comparator architectures [6], neural networks [7], and, more recently, sorting data structures [8]. The aim of this paper is to present an algorithm for the S-PWL computation based on the decomposed formulation of the simplicial model. The main contribution of this work is the procedure for determining the *N*-dimensional hypercube path in a simplicial decomposition. The paper is organized in the following way: In Section 2, a general scope of the S-PWL is given; Section 3 deals with the decomposed S-PWL representation; the evaluation algorithm is presented in Section 4, and an illustrative example is described in Section 5.

2. THE SIMPLICIAL PWL FUNCTION

In this section, a brief description of the simplicial piecewise-linear function (S-PWL) is approached. For further details about this kind of representation, the reader is referred to [9]. Let us consider, for simplicity, a two dimensional domain. If a boundary configuration is defined so that the entire domain is partitioned into simplices and a function value is associated to each intersection (given by a vertex), as illustrated by figure 1, then it is possible to define a PWL function with the following characteristics [10].

a) Considering the function values assigned to each vertex, a unique linear (local) function is defined for each simplex.
b) The different linear (local) functions that are continuous on the boundaries of the partition define a continuous PWL function.

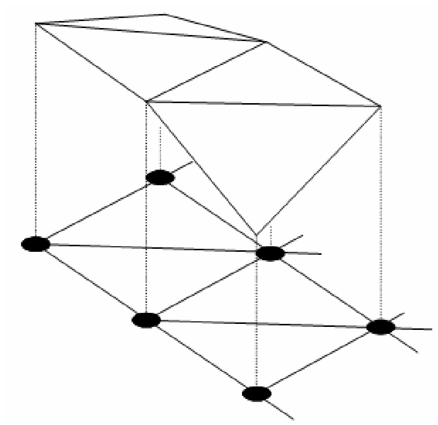

Figure 1. Constructive approach in a two dimensional S-PWL function.

The extension of this idea to a *N*-dimensional domain leads us to define simplices of $N+1$ vertices. Then, if one function is associated with each vertex, it is possible to uniquely determine a linear (local) function for each simplex. In this way, a continuous PWL function is determined by the collection of all the local functions. From this procedure, it is clear that any arbitrary PWL function defined over the simplicial boundary configuration introduced is uniquely determined by its values on the vertices. In reference [11], this function constructive approach is used and it is stated that a PWL function defined over a *N*-dimensional simplicial domain can be expressed by the weighted sum:

$$\boldsymbol{F}(\mathbf{X})=\sum_{k=1}^{N+1} c_k \mu_k \qquad (1)$$

Where *N* is the *N*-th dimension, μ_k are internal parameters, and c_k are the values of the function at the boundary vertices of the simplex where $\mathbf{X}$ is located. In order to gain insight in the geometrical meaning of those variables and parameters, Figure 2 depicts a two dimensional PWL function with a (3×2)-grid. In this figure, the (local) representation of $\boldsymbol{F}(\mathbf{X})$ at the $\{(1,1),(1,2),(2,2)\}$-simplex is considered. Notice that the triangular shadowed region over the simplicial partition indicates the simplex where $\mathbf{X}$ is found, and $\{c_{11}, c_{12}, c_{22}\}$ is the set of values of $\boldsymbol{F}(\mathbf{X})$ at the simplex vertices $\{v_{11}, v_{12}, v_{22}\}$.

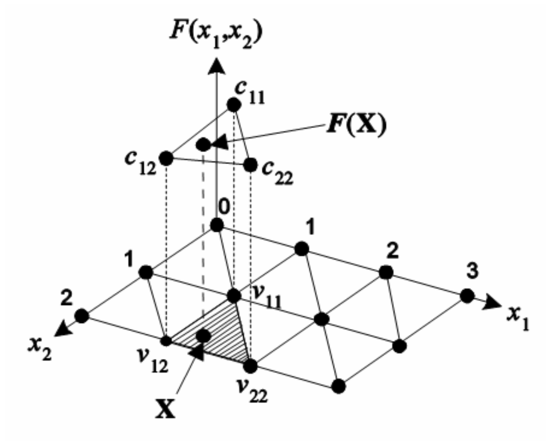

Figure 2. Two dimensional S-PWL function at the{(1,1),(1,2),(2,2)}-simplex.

In accordance with Eq.1, $c_1 = c_{22}$, $c_2 = c_{12}$, and $c_3 = c_{11}$. The μ-parameters result from the simplicial decomposition of $\mathbf{X}$. That type of decomposition is reported in [12], and it states that a point $\mathbf{X}$ belonging to a *N*-dimensional simplicial domain can be decomposed as

$$\mathbf{X} = \sum_{k=1}^{N+1} \mu_k \mathbf{v}_k \qquad (2)$$

with μ-parameters fulfilling the relation

$$1 = \sum_{k=1}^{N+1} \mu_k \qquad (3)$$

Where $\mathbf{v}_k$ are the vertices of each simplex, and its consecutive sequence of appearing in Eq.2 describes and *N*-dimensional hypercube path. In order to clarify the simplicial decomposition concept, let us consider an example.

Example 1: Let $\mathbf{X} = (1.3,5.6,2.1)$ be a point in a $\Re^3$ simplicial domain. This point can be decomposed as follows:

1) Firstly, $\mathbf{X}$ is decomposed into integer ($\mathbf{X}_{\text{int}}$) and fractional ($\mathbf{X}_{frac}$) parts:

   $\mathbf{X}_{\text{int}} = [1,5,2]$, and $\mathbf{X}_{frac} = [0.3,0.6,0.1]$

2) Then, a searching of minimum element and pick process is repeated *N* times as follows:

   2.1) The minimum $\mathbf{X}_{frac}$ is picked (i.e. 0.1), and it is decomposed as

   $$\mathbf{X}_{frac} = 0.1 \times [1,1,1] + [0.2,0.5,0]$$

   2.2) The minimum element of $[0.2,0.5,0]$ is picked (i.e. 0.2), and it is decomposed as $\mathbf{X}_{frac} = 0.1\times[1,1,1] + 0.2\times[1,1,0] + [0,0.3,0]$

2.3) The minimum element of $[0,0.3,0]$ is picked (i.e. 0.3), and it is decomposed as $\mathbf{X}_{frac} = 0.1\times[1,1,1] + 0.2\times[1,1,0] + 0.3\times[0,1,0]$

3) After that, the decomposition of $\mathbf{X}_{frac}$ is completed by using Eq.3 as

$\mathbf{X}_{frac} = 0.1\times[1,1,1] + 0.2\times[1,1,0] + 0.3\times[0,1,0] + 0.4\times[0,0,0]$

Where $0.4 = 1-(0.1+0.2+0.3)$

4) Finally, $\mathbf{X}$ can be expressed as

$$\mathbf{X} = \begin{bmatrix}1\\5\\2\end{bmatrix} + 0.1\begin{bmatrix}1\\1\\1\end{bmatrix} + 0.2\begin{bmatrix}1\\1\\0\end{bmatrix} + 0.3\begin{bmatrix}0\\1\\0\end{bmatrix} + 0.4\begin{bmatrix}0\\0\\0\end{bmatrix}$$

The vertex sequence $[1,1,1]\to[1,1,0]\to[0,1,0]\to[0,0,0]$ indicates the hypercube path depicted in Figure 3.

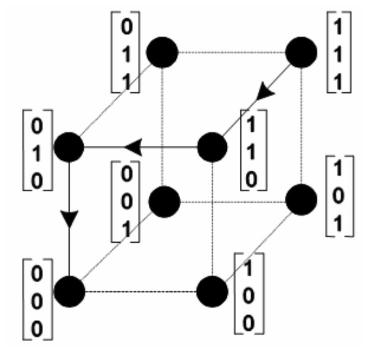

Figure 3. Hypercube path for Example 1.

From this example: $\mu_1 = 0.4$, $\mu_2 = 0.3$, $\mu_3 = 0.2$, and $\mu_4 = 0.1$. It is important to point out that the hypercube path is implicitly the core of Eq.1 because it establishes the (μ_k, c_k)-relation. A second example will make it easier to understand.

Example 2: Consider $\boldsymbol{F}(\mathbf{X})$ be a PWL function defined over a $\Re^3$ simplicial domain. Also, suppose this function is described by a set of $c_{\hat{\mathbf{V}}_k}$ -values which denote the value of $\boldsymbol{F}(\mathbf{X})$ at the vertices $\hat{\mathbf{V}}_k$, and the $\hat{\mathbf{V}}$ -vertices are given by $\hat{\mathbf{V}}_k = \mathbf{X}_{\text{int}} + \mathbf{v}_k$, for $k = 1,2,\cdots,(N+1)$.
Determine the form of Eq.1 in order to compute $\mathbf{X} = (1.3,5.6,2.1)$.

Taking into account the simplicial decomposition of example 1, the following results are used: $\mathbf{X}_{\text{int}} = [1,5,2]$, $[1,1,1] \to [1,1,0] \to [0,1,0] \to [0,0,0]$ as hypercube path, and $\{\mu_1 = 0.4, \mu_2 = 0.3, \mu_3 = 0.2, \mu_4 = 0.1\}$.

From hypercube path: $v_1 = [0,0,0]$, $v_2 = [0,1,0]$, $v_3 = [1,1,0]$, and $v_4 = [1,1,1]$.

Eq.1 takes the form: $\boldsymbol{F}(\mathbf{X}) = 0.1 \times c_4 + 0.2 \times c_3 + 0.3 \times c_2 + 0.4 \times c_1$

The $\hat{\mathbf{V}}$ -vertices are computed as follows:

$\hat{\mathbf{V}}_4 = \mathbf{X}_{\text{int}} + \mathbf{v}_4 = [1,5,2] + [1,1,1] = [2,6,3]$, $\hat{\mathbf{V}}_3 = \mathbf{X}_{\text{int}} + \mathbf{v}_3 = [1,5,2] + [1,1,0] = [2,6,2]$,

$\hat{\mathbf{V}}_2 = \mathbf{X}_{\text{int}} + \mathbf{v}_2 = [1,5,2] + [0,1,0] = [1,6,2]$, $\hat{\mathbf{V}}_1 = \mathbf{X}_{\text{int}} + \mathbf{v}_1 = [1,5,2] + [0,0,0] = [1,5,2]$.

Because of $c_k = \hat{\mathbf{V}}_k$ for $k = 1,2,\cdots,(N+1)$, then $\boldsymbol{F}(\mathbf{X})$ is rewritten as

$\boldsymbol{F}(\mathbf{X}) = 0.1 \times c_{[2,6,3]} + 0.2 \times c_{[2,6,2]} + 0.3 \times c_{[1,6,2]} + 0.4 \times c_{[1,5,2]}$

3. DECOMPOSED S-PWL REPRESENTATION

A model description is denoted as decomposed if their constitutive independent and dependent variables can be collected separately in a system of equations. This kind of models have demonstrated advantages over explicit and implicit representations in the areas of modeling and analysis [13], [14]. If Eq, (1), (2), and (3) are arranged into a matrix form, the decomposed system of Eq. 4 is obtained.

$$\begin{bmatrix} \boldsymbol{F}(\mathbf{X}) \\ \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_N \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{bmatrix} \quad (4)$$

A sub-system of equations can be taken from the decomposed representation and recast as

$$\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ 1 & 1 & \cdots & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{bmatrix} \quad (5)$$

After solving μ_k from Eq.5, $\boldsymbol{F}(\mathbf{X})$ can be expressed by

$$\boldsymbol{F}(\mathbf{X}) = \begin{bmatrix} c_1 & c_2 & \cdots & c_N \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ 1 & 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \quad (6)$$

From Eq.6 it can be seen that the evaluation of $\mathbf{X}$ at any arbitrary S-PWL function $\boldsymbol{F}(\mathbf{X})$ is determined by the values on the vertices.

4. EVALUATION ALGORITHM

Consider a S-PWL $\boldsymbol{F}(\mathbf{X})$ defined over a $\Re^N$ domain. Assume the values on the vertices are collected into an *N*-dimensional matrix $\mathbf{M}_D$ where the index of each element of the matrix corresponds with the simplicial vertex $\mathbf{v}_i$. An evaluation algorithm for the decomposed system of Eq.6 is summarized in the following steps:

1. Input-Data: Let $\mathbf{X} = [x_1, x_2, \cdots, x_N]$ be the *N*-dimensional input for the $\Re^N$ S-PWL function.
2. Decomposition: Each x_j element of $\mathbf{X}$ (for j=1…*N*) is decomposed by integer and fractional part, here expressed by the notation $x_j = x_{\text{int}_j}.x_{frac_j}$. All the x_{int_j} and x_{frac_j} elements are grouped into the sets $\mathbf{X}_{\text{int}} = [x_{\text{int}_1}, x_{\text{int}_2}, \cdots, x_{\text{int}_N}]$ and $\mathbf{X}_{frac} = [x_{frac_1}, x_{frac_2}, \cdots, x_{frac_N}]$, respectively.
3. Sorting: Let $X_{sorted} = \lfloor x_{s_1}, x_{s_2}, \cdots, x_{s_N} \rfloor$ be a vector which includes the x_{frac_j} elements sorted by the relation $x_{s_1} < x_{s_2} < \cdots < x_{s_N}$.
4. Hypercube path: The *N*-dimensional hypercube path is determined by the following procedure:

   4.1 Define the $N \times N$ matrices: $\mathbf{M}_F$, and $\mathbf{M}_S$ as

$$\mathbf{M}_F = \begin{bmatrix} x_{frac_1} & \cdots & x_{frac_1} \\ x_{frac_2} & \cdots & x_{frac_2} \\ \vdots & & \vdots \\ x_{frac_N} & \cdots & x_{frac_N} \end{bmatrix} \qquad \mathbf{M}_S = \begin{bmatrix} x_{s_1} & x_{s_2} & \cdots & x_{s_N} \\ \vdots & & & \vdots \\ x_{s_1} & x_{s_2} & \cdots & x_{s_N} \end{bmatrix}$$

   4.2 Obtain the $N \times N$ matrix $\mathbf{M}_T$: $\quad \mathbf{M}_T = \mathbf{M}_F - \mathbf{M}_S$

   4.3 Compute the $N \times N$ matrix $\mathbf{M}_I$ by the following procedure:

```
for $i$ from 1 to $N$ do
    for $j$ from 1 to $N$ do
        if $\mathbf{M}_T[i, j] \neq 0$ then $\mathbf{M}_I[i, j] = 1$
        else $\mathbf{M}_I[i, j] = 0$
    end for
end for
```

```
            for $i$ from $2$ to $N$ do
                for $j$ from 1 to $N$ do
                        $\mathbf{M}_I[j,i] = \mathbf{M}_I[j,i-1] \times \mathbf{M}_I[j,i]$
                end for
            end for
    4.3 Form the $N \times (N+1)$ $\mathbf{M}_H$ matrix by joining the $N \times 1$ $\mathbf{EC}$-vector to $\mathbf{M}_I$.
```

$$\mathbf{EC} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \mathbf{M}_H = [\mathbf{EC} \ \vdots \ \mathbf{M}_I]$$

5. Addressing: Determine the index connection between c_i-parameters and the $\mathbf{M}_D$-elements by

$$\hat{\mathbf{v}}_i = \begin{bmatrix} x_{\text{int}_1} \\ x_{\text{int}_2} \\ \vdots \\ x_{\text{int}_N} \end{bmatrix} + \begin{bmatrix} \mathbf{M}_H[1,i] \\ \mathbf{M}_H[2,i] \\ \vdots \\ \mathbf{M}_H[N,i] \end{bmatrix}, \qquad c_{N+1-i} = \mathbf{M}_D[\hat{v}_{i+1}] \qquad \text{For } i = 0,1,\cdots,N.$$

6. Evaluation: Substitute the c_i-values determined in the previous step and compute $\boldsymbol{F}(\mathbf{X})$ as follows:

$$\boldsymbol{\mu} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_N \\ 1 & 1 & \cdots & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}_{frac}{}^{T} \\ 1 \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_N \end{bmatrix}, \quad \boldsymbol{F}(\mathbf{X}) = [c_1 \quad c_2 \quad \cdots \quad c_N][\boldsymbol{\mu}]$$

5. EXAMPLE

Consider the PWL function $\boldsymbol{F}(x_1, x_2)$ depicted in Figure 4. This function is defined over a domain partitioned into 18 simplices in a (3×3) grid.

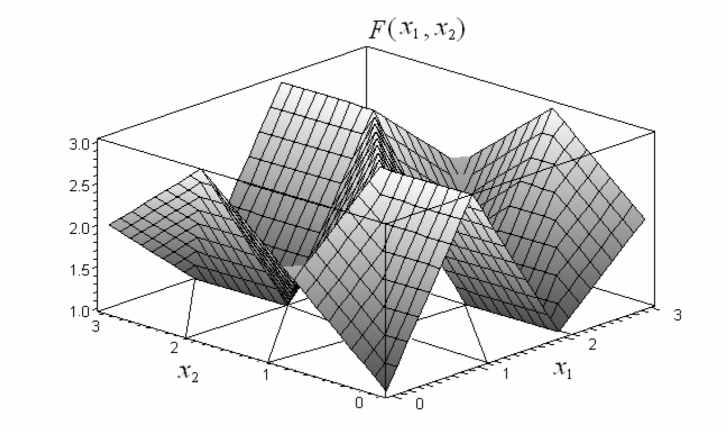

Figure 4. Two dimensional S-PWL function of example

Table I shows the values of $\boldsymbol{F}(x_1, x_2)$ at the vertices.

$\mathbf{v}_k$ -Vertex	0,0	1,0	2,0	3,0	0,1	1,1	2,1	3,1	0,2	1,2	2,2	3,2	0,3	1,3	2,3	3,3
$\boldsymbol{F}(\mathbf{v}_k)$	1	3	1	2	2	3	1	3	3	1	3	2	2	1	3	1

Table I. Values of at the vertices of example.

Those values are collected into the $\mathbf{M}_D$ matrix.

$$\mathbf{M}_D == \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 2 \\ 3 & 3 & 1 & 1 \\ 1 & 1 & 3 & 3 \\ 2 & 3 & 2 & 1 \end{bmatrix}$$

In this example, the evaluation of an arbitrary input point, for example, $\mathbf{X} = (1.8, 2.1)$ is achieved in accordance with the algorithm described in the previous section. Notice that $n = 2$, and the following data are obtained:

Step 1: Input-Data.

$$\mathbf{X} = [x_1, x_2] = [1.8, 2.1]$$

Step 2: Decomposition.

$$X_{\text{int}} = \lfloor x_{\text{int}_1}, x_{\text{int}_2} \rfloor = [1,2], X_{frac} = \lfloor x_{frac_1}, x_{frac_2} \rfloor = [0.8, 0.1]$$

Step 3: Sorting.

$$X_{sorted} = \lfloor x_{frac_1}, x_{frac_2} \rfloor = [0.1, 0.8]$$

Step 4: Hypercube path.

$$\mathbf{M}_F = \begin{bmatrix} 0.8 & 0.8 \\ 0.1 & 0.1 \end{bmatrix} \qquad \mathbf{M}_S = \begin{bmatrix} 0.1 & 0.8 \\ 0.1 & 0.8 \end{bmatrix}, \ \mathbf{M}_T = \mathbf{M}_F - \mathbf{M}_S = \begin{bmatrix} 0.7 & 0 \\ 0 & -0.7 \end{bmatrix}$$

After the procedure:

First $\mathbf{M}_I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, then $\mathbf{M}_I = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$, $\mathbf{EC} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\mathbf{M}_H = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

Step 5: Addressing**.**

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \ \mathbf{v}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{v}_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\hat{\mathbf{v}}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \ \hat{\mathbf{v}}_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, \ \hat{\mathbf{v}}_3 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$c_3 = \mathbf{M}_D[\hat{v}_1] = \mathbf{M}_D[2,3] = 3, \ \ c_2 = \mathbf{M}_D[\hat{v}_2] = \mathbf{M}_D[2,2] = 3, \ \ c_1 = \mathbf{M}_D[\hat{v}_3] = \mathbf{M}_D[1,2] = 1$$

Step 6: Evaluation*.*

$$\boldsymbol{\mu} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}_{frac} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0.8 \\ 0.1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.7 \\ 0.2 \end{bmatrix} = \begin{bmatrix} \mu_3 \\ \mu_2 \\ \mu_1 \end{bmatrix}$$

$$\boldsymbol{F}(\mathbf{X}) = [c_3 \quad c_2 \quad c_1][\boldsymbol{\mu}] = 0.1 \times 3 + 0.7 \times 3 + 0.2 \times 1 = 2.6$$

6. CONCLUSIONS

An algorithm for the evaluation of arbitrary N-dimensional S-PWL functions was presented. It includes an effective procedure for determining the N-dimensional hypercube path. In accordance with the algorithm, the value of the function is computed by its values on the vertices, and the vertices are inherently immersed into the hypercube path which results from the simplicial decomposition of the evaluation point. Also, it can be seen that in order to make the algorithm more effective, the process of computing -parameters can be improved because the inverse operation is done over a matrix which only contains zero and one elements, and it also has a regular structure (i.e. the first column is and the ending column is always).

7. REFERENCES

[1] Chua L.O. & Ying L.P., Finding all solutions of piecewise-linear circuits, International Journal of Circuit Theory and Applications, Vol. 10, July, 1982, pp. 201-229.

[2] Yamamura K. & Machida A., Finding all solutions of piecewise-linear resistive circuits with high approximation accuracy, J. Circuit Syst. Comput., Vol. 15, No.3, June, 2006, pp. 389-398.

[3] Leenaerts D.M.W. & Hegt J.A., Finding all solutions of piecewise-linear functions and application to circuits design, International Journal of Circuit Theory and Applications, Vol. 19, No. 2, 1991, pp. 107-123.

[4] Julian P., Dogaru L. & Chua L.O., A piecewise-linear simplicial coupling cell for CNN gray-level image processing, Circuit and Systems I: Fundamental Theory and Applications, Vol. 49, No. 7 July, 2002, pp. 904-913.

[5] Castro L., Figueroa J. & Agamennoni O., BIBO stability for NOE model structure using HL-CPWL functions, Proceedings of MIC.05-IASTED, 2005, Innsbruck Austria.

[6] Di Federico M., Julian P., Poggi T. & Storace M., A Simplicial PWL Integrated Circuit Realization, Proceedings of IEEE International Symposium on Circuits and Systems, 2007, pp. 685-688, New Orleans, LA USA , May.

[7] Sun X. & Wang S., A Special Kind of Neural Nertworks: Continuous Piecewise Linear Functions, LNCS Advances in Neural Networks, 2005, pp. 375-379.

[8] Jimenez-Fernandez V.M., Rodríguez J.A., Julián P., Agamennoni O. & Di Federico M., Digital Architecture for R6 PWL Function Computation, Actas de la Escuela Argentina de Microelectrónica, Tecnología y Aplicaciones, 2007, pp. 1-6, Córdoba Arg., September.

[9] Julián P., A High Level Canonical Piecewise-Linear Representation: Theory and Applications, Doctoral Thesis, Universidad Nacional del Sur, B. Blanca Argentina, 1998.

[10] Julián P & Agamennoni O., High Level Canonical Piecewise-Linear Representation Using a Simplicial Partition, IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 46, No. 4, April, 1999, pp. 463-480.

[11] Julián P & Chua L.O., A Piecewise-Linear Simplicial Coupling Cell for CNN Gray-Level Image Processing, IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 49, No. 7, July, 2002, pp. 904-913.

[12] Chien M. & Kuh E., Solving Nonlinear Resistive Nerworks Using Piecewise-Linear Analysis and Simplicial Subdivision, IEEE Transactions on Circuits and Systems-I, Vol. CAS 24, No. 7, June, 1977, pp. 305-317.

[13] Jiménez-Fernández V.M., Decomposed Piecewise-Linear Representation Applied to DC Analysis, Doctoral Thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica INAOE, Santa María Tonantzintla Puebla, México, 2006.

[14] Brzobohaty J. & Kolka Z., Decomposed Parametric Form of the State Model of a Piecewise-Linear System, IEEE International Symposium on Circuits and Systems,Vol. 6, 1994, pp. 25 – 28, June.

Authors' Biography

Víctor Manuel Jiménez Fernández

Was born in Puebla, Mexico on December 5th 1974. He received a BSc degree in electronics engineering from the Instituto Tecnológico de Veracruz in 1998, the M.S. degree from the Universidad de las Américas-Puebla in 2000, and the Ph.D. degree from the Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) Puebla, Mexico in 2006. From 2006 to 2007 he was a visiting researcher in the Departamento de Ingeniería Eléctrica y de Computadoras at the Universidad Nacional del Sur, Bahía Blanca, Argentina. Since February 2008 he has been an assistant researcher at the INAOE-Mexico.

Juan Agustín Rodríguez

Was born in the city of Bahía Blanca, Argentina, on May 6th, 1981. He received a BSc degree in computer systems engineering in 2007 from Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina. He currently is a PhD student in Computer Science at UNS and he has a (Type I) grant from CONICET. His research interests include computer architectures, scientific computing, VLSI design and electronic design automation.

Pedro Marcelo Julián

Was born in the city of Bahía Blanca, Argentina, on June 28, 1970. He received a BSc degree in electrical engineering in 1994 and the PhD degree in control system in 1999, both from Universidad Nacional del Sur (UNS). From 2000 to 2002 he was a visiting scholar in the Nonlinear Electronics Laboratory of the University of California at Berkeley, USA. From 2002 to 2003 he was a visiting scholar in the Sensory Communication & Microsystems Laboratory of the Johns Hopkins University, Baltimore, USA. He has been an Assistant Professor in the Departamento de Ingeniería Eléctrica y Computadoras at UNS since 2003. His research interests areas are computation, integrated circuits and systems theory/ design.

Osvaldo Enrique Agamennoni

Was born in Bahía Blanca, Argentina on December 21, 1953. He received a BSc degree in electrical engineering and the doctorate in Control System, both from the Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina, in 1979 and 1991, respectively. From 1980 to 1983 he worked in electronic circuits design. From 1992 to 1994 he was a postdoctoral fellow in the Chemical Engineering Department of the University of Sydney. He currently is Profesor Titular in the Departamento de Ingeniería Eléctrica y de Computadoras at the Universidad Nacional del Sur. His research interests include nonlinear system modeling, identification and control of nonlinear system.