International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tetn20

Optimum piecewise-linear transcoders Part 1. Weighted minimax piecewise-linear approximation and minimax decomposition of piecewise functions

YANNICK DEVILLE [a]

[a] Laboratoires d'Electronique Philips , 22, Avenue Descartes, BP 15, Limeil-Brévannes, Cedex, 94453, France.

Published online: 19 Apr 2007.

To cite this article: YANNICK DEVILLE (1994) Optimum piecewise-linear transcoders Part 1. Weighted minimax piecewise-linear approximation and minimax decomposition of piecewise functions, International Journal of Electronics, 77:6, 823-844, DOI: 10.1080/00207219408926104

To link to this article: http://dx.doi.org/10.1080/00207219408926104

INT. J. ELECTRONICS, 1994, VOL. 77, NO. 6, 823-844

Optimum piecewise-linear transcoders
Part 1. Weighted minimax piecewise-linear approximation and minimax decomposition of piecewise functions

YANNICK DEVILLE†

This paper mainly concerns the following mathematical problem: an initial single-argument single-valued function F is known only through a set of points $P_i(X_i, Y_i, W_i)$, with $Y_i = F(X_i)$, and with application-dependent weights W_i. An optimum (or almost-optimum in some cases) approximating function f should be derived from these points P_i. f is searched within a predefined class of functions. Various such classes are successively considered. They consist of subsets of piecewise-linear functions. The approximation criterion used to derive f from points P_i consists of determining an approximating function which minimizes an overall error. This error is typically defined as the maximum among local weighted errors associated with each point P_i. Beyond piecewise-linear approximation, this paper also presents algorithms for optimizing the domains of operation of the subfunctions of any type of piecewise function acccording to a possibly-weighted minimax criterion. This investigation is motivated by an industrial application, i.e. automatic TV tuner alignment. This application is outlined in this paper and detailed in a companion paper (see Deville 1994a), which shows that the proposed approach applies to a wide class of systems, including active filters and phase shifters.

1. Introduction

This paper mainly deals with the type of mathematical problem which may be summarized as follows (the exact definitions are provided in the subsequent sections): an initial single-argument single-valued function F is known only through a set of points $P_i(X_i, Y_i, W_i)$, where X_i and Y_i are real values with $Y_i = F(X_i)$, and where the weights W_i may be set to any real positive values depending on the considered application. An optimum (or almost-optimum in some cases) approximating function f should be derived from this set of points P_i, according to the approximation criterion provided at the end of this section. This function f is searched within a predefined class of functions. The class considered roughly consists of piecewise-linear (PWL) functions. This, in fact, yields various cases, depending on the exact definition of the considered class of functions.

These cases are studied successively, according to increasing complexity, so this paper is organized as follows. Section 2 presents the basic case when the approximating function is restricted to consist of a single line. Sections 3 and 4 concern the case when the approximating function is PWL and allowed to have discontinuous segments. This investigation is split in two, depending on whether the 'interpolation capability' of the approximating functions is ignored (§ 3) or taken into account (§ 4). The algorithms developed in § 3 and 4 in fact have wider applicability than the PWL approximation. Therefore, they are presented in a general context and their specific application to PWL functions is provided at each step of the approach. Then, § 5

Received 18 April 1994; accepted 20 May 1994.

†Laboratoires d'Electronique Philips, 22, Avenue Descartes, BP 15, 94453 Limeil-Brévannes Cedex, France.

presents the case when the approximating function is PWL and, in addition, is required to be continuous. Section 6 corresponds to the situation when the approximating function is continuous and PWL, defined by a set of points (i.e. the limit points between its adjacent segments and one additional point in each extreme segment) and when the coordinates of these points may only have integral values. Finally, §7 contains a summary of all the results thus obtained and information about related problems. This section also outlines the industrial TV tuner application which has motivated these investigations. It thus justifies all the steps of the approach used in this paper.

The approximation criterion used to determine an optimum approximating function typically corresponds to a weighted minimax approach. More precisely, first consider the case when the class of approximating functions consists of possibly-discontinuous PWL functions (and when their 'interpolation capability' is ignored). In this case, a local weighted error is defined between any point P_i corresponding to the initial function F and any approximating function f (as explained in §2 and 3, this error is equal to the absolute value of the difference between the values of F and f for $X = X_i$, multiplied by the weight W_i of P_i). Then, the overall error between the complete set of points P_i and f is defined as the maximum among the local weighted errors corresponding to all points P_i. Finally, the approximation criterion used in this paper for a given set of points P_i consists of determining an approximating function f, which minimizes this overall error, i.e. which minimizes the maximum among all local weighted errors. Hence the name 'weighted minimax' used for this approximation criterion. This criterion corresponds to a worst-case approach: the overall performance of an approximating function is defined by the worst local error that it yields, i.e. this function should provide a low local error for all the input data points. The selection of this criterion is motivated by the application outlined in §7.

The other cases treated in this paper yield slightly different approximation criteria, but they also correspond to (weighted) minimax approaches. Therefore, within the overall domain of PWL functions, the specific features of this paper consist of performing an approximation with weighted minimax criteria, and in only considering single-argument single-valued functions. This should first be contrasted with standard linear approximation methods, such as stright-line Least-Mean-Square (LMS) approximation (see Papoulis 1984, Box *et al.* 1978): the latter approach aims at providing a low average quadratic error and most often yields higher local errors for some points P_i. This is not acceptable in the application that we consider, where overall performance is defined by the worst local error and is not optimized by this standard approach. This paper should also be distinguished from the investigations of PWL functions, which are particularly motivated by applications to nonlinear circuits or system modelling: these investigations relate to higher dimensional spaces, they are concerned with the existence of approximating functions rather than their practical determination according to given criteria (see Lin and Unbehauen 1992) or they use other approximation criteria (see Batruni 1991) and/or they apply to specific initial functions (see Yamamura 1992), or they concern functional representation instead of approximation (see in particular Kahlert and Chua 1990, 1992 and the review provided by Kevenaar and Leenaerts 1992).

The algorithms described below are derived from various theorems. These theorems are provided in this paper, but their proofs are skipped for the sake of brevity (details are given by Deville 1994 b).

2. Weighted minimax linear approximation

2.1. *Problem statement*

Input data. The initial single-argument single-valued function F is known only through a set S containing n points P_i (see Fig. 1), with $i=0$ to $(n-1)$. Each such point has two main real coordinates, denoted X_i and Y_i, with $Y_i=F(X_i)$. In addition, an arbitrary weight W_i is associated with each point, as explained below. The following conditions are set on these coordinates: each weight W_i should be strictly positive and the points should be ordered so that X_i increases strictly with the point index i (the latter condition is set only because it yields simplified expressions of the proposed algorithms).

Class of approximating functions. In this section, this class contains all functions f defined as: $f(x)=px+q$, where p and q may have any real values. Each such function corresponds to a line D having a slope p and an intercept q. The term 'line' and notation D are used below to represent f.

Definition 1—Point/line error E_{PL}: The error between an input data point P_i and a line D, defined by (p,q), is defined as

$$E_{PL}(P_i, D)=W_i.|F(X_i)-f(X_i)|=W_i.|Y_i-(p.X_i+q)| \tag{1}$$

□

The basic term of this error is the absolute value of the difference between the initial and approximating functions for the considered X coordinate. Moreover, this term is multiplied by the weight W_i. The selection of these weights allows one to rescale individually the basic error terms corresponding to various X coordinates according to the considered application. In particular, the weights required for TV tuners are provided by Deville (1994 a).

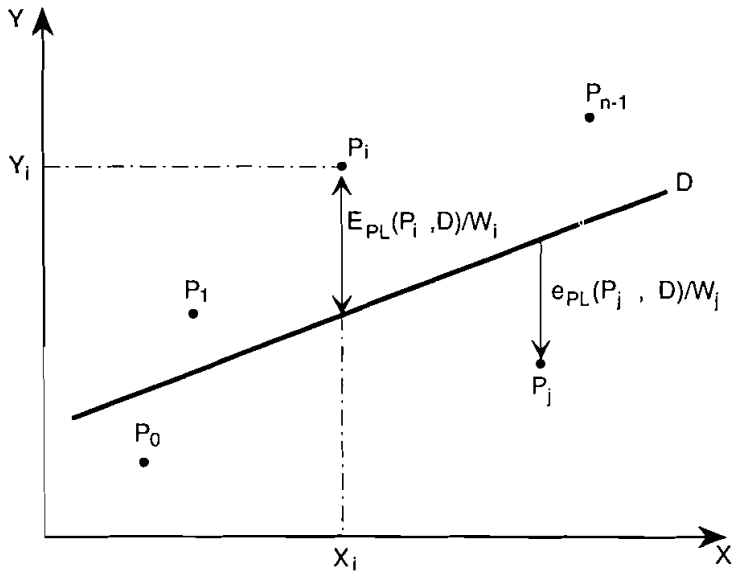

Figure 1. Weighted minimax linear approximation: input data set S containing n points P_i, with $i=0$ to $(n-1)$, line D and resulting errors and gaps.

Definition 2—Set/line error E_{SL}: The error between an input data set S of points P_i, with $i=0$ to $(n-1)$, and a line D defined by (p, q) is defined as

$$E_{SL}(S, D) = \max_{i \in \{0 \ldots (n-1)\}} [E_{PL}(P_i, D)] \tag{2}$$

□

As explained in § 1, this definition corresponds to a worst-case approach.

Definition 3—Optimum lines and optimum error: Given an input data set S, the optimum lines are all the lines D_{opt} which minimize the above-defined error $E_{SL}(S, D)$, among all the existing lines, i.e. among all real values p and q. The optimum error $E_{S_{opt}}(S)$ is the error $E_{SL}(S, D_{opt})$ common to all optimum lines. □

Goal of the investigation, approximation criterion. Given an input data set S, this section aims at determining some corresponding optimum line. The associated minimum error is also determined. As explained in the previous section, this correspnds to a weighted minimax approach, as opposed to LMS approximation.

2.2. *Solutions for input data sets containing 1 or 2 points*

When the input data set S contains $n=1$ or 2 points, the solutions to the problem defined above have specific expressions. These special cases are presented here. In the remainder of this section, it is assumed that $n \geqslant 3$.

Theorem 1: *If the input data set S* contains $n=1$ *points, there is an infinity of optimum lines; i.e. all the lines which contain this point. The optimum error is:* $E_{S_{opt}}(S)=0$.

Theorem 2: *If the input data set S contains $n=2$ points, there is exactly one optimum line. This is the line which contains these points. The optimum error is:* $E_{S_{opt}}(S)=0$.

2.3. *Performance criteria*

Several algorithms for solving the above-defined problem are presented in the four next subsections. The overall performance of any of them is measured by the set of parameters defined in this subsection.

Complexity. The complexity of the proposed algorithms is defined with respect to their basic action, which consists of computing an error E_{PL} (or gap e_{PL} defined below). This yields two classes of algorithms. Some algorithms have a complexity which is 'independent of input data', i.e. for a given number n of input data points, the number of errors E_{PL} computed to obtain the solution does not depend on the coordinates of these points. This number of errors is then proportional to a certain power of n, which defines the complexity of the algorithm. Conversely, the number of errors E_{PL} computed for the other algorithms 'depends on input data', i.e. on point coordinates. The lowest and highest possible values of this number of errors then correspond to different powers of n. They define the range of complexity of such algorithms.

Sensitivity. The approach presented below consists of first deriving initial algorithms from mathematical theorems. These algorithms always provide the exact solution to the considered problem when an infinite precision is available to represent numbers and to perform computations. However, the final algorithms are to be executed on computers, which use a finite precision. Therefore, the initial

algorithms are then modified in order to take into account truncation effects. Some of the resulting algorithms are insensitive to truncations, i.e. when run on a computer they always provide the solution. The others are almost-insensitive to truncations, i.e. they only fail for very special data.

Accuracy. Some practical algorithms provide an 'almost-exact' solution (i.e. the exact one except that it is rounded with the precision of the computer). The other algorithms only provide an 'approximated' solution (i.e. a solution which differs from the exact one by an amount which is higher than plain truncation effects, but which is selected by the user).

Anyone of the proposed algorithms only has a part of the desired features. To obtain all these features, combinations of these algorithms are proposed at the end of this section.

2.4. *First approach based on triplets of input data points*

This subsection and the next ones contain two parts. First, the required definitions and theorems are provided. Then, the resulting algorithms are presented.

Definition 4—Point/line gap e_{PL}: The gap between an input data point P_i and a line D, defined by (p, q), is defined as

$$e_{PL}(P_i, D) = W_i.[F(X_i) - f(X_i)] = W_i.[Y_i - (pX_i + q)] \tag{3}$$

□

This gap e_{PL} is the signed counterpart of the error E_{PL} defined above (see Fig. 1). Its sign defines the relative positions of P_i and D (e.g. P_i is above D when $e_{PL}(P_i, D) > 0$).

Theorem 3: *If $\{P_i, P_j, P_k\}$ is a triplet of points of the input data set S, with $X_i < X_j < X_k$, then exactly one line D meets the following conditions*

$$(a) \quad e_{PL}(P_i, D) = -e_{PL}(P_j, D) \tag{4}$$

$$(b) \quad e_{PL}(P_i, D) = e_{PL}(P_k, D) \tag{5}$$

Moreover, the error E_{PL} between a point of this triplet and this line is the same for all three points of this triplet: $E_{PL}(P_i, D) = E_{PL}(P_j, D) = E_{PL}(P_k, D)$.

The slope and intercept of this line are easily derived from the linear system (4)-(5).

Definition 5: If $\{P_i, P_j, P_k\}$ is a triplet of points of the input data set S, with $X_i < X_j < X_k$, then the line defined by Theorem 3 is called *the line associated with this triplet*. Moreover, the common error E_{PL} between any point of the triplet and the line associated with this triplet is called *the error associated with this triplet* and is denoted by E_T: $E_T(P_i, P_j, P_k) = E_{PL}(P_j, D) = E_{PL}(P_j, D) = E_{PL}(P_k, D)$. □

Theorem 4: *If the input data set S contains $n \geqslant 3$ points, there is exactly one optimum line. It is the only line D such that there exists at least one triplet $\{P_i, P_j, P_k\}$ of points of S, with $X_i < X_j < X_k$, such that the following conditions are met*

$$(a) \quad D \text{ is the line associated with this triplet}$$

$$(b) \quad \forall P_t \in S - \{P_i, P_j, P_k\}, \quad E_{PL}(P_t, D) \leqslant E_T(P_i, P_j, P_k) \tag{6}$$

A first algorithm providing the solution to the considered problem may be deduced from Theorem 4 (see Fig. 2). As a result of Theorem 4, this algorithm consists of successively using various triplets of points $\{P_i, P_j, P_k\}$ of S, with $X_i < X_j < X_k$. These triplets may be used in any order. If some *a priori* knowledge on the input data is available, it should be exploited so as first to use the triplets which have a higher probability of providing the solution (in order to increase speed). Otherwise, the loops on i, j, k of Fig. 2 may be used. For each such triplet, the associated line $D(P_i, P_j, P_k)$ and error $E_T(P_i, P_j, P_k)$ are first determined. Then, the algorithm tests (again in 'any' order) if the desired condition $E_{PL}[P_l, D(P_i, P_j, P_k)] \leqslant E_T(P_i, P_j, P_k)$ (see (6)) is met for each of the points P_l of S, except for the points of the triplet (for which it is met by construction, but this would often fail in practice due to rounding errors). If this condition is not met for at least one point, Theorem 4 states that the current triplet does not provide the optimum. Then, another triplet is considered (i.e. go to 'next k' in the loop on k in Fig. 2). Otherwise, if $E_{PL}[P_l, D(P_i, P_j, P_k)] \leqslant E_T(P_i, P_j, P_k)$ is met for all the other points of S, the current triplet provides the solution. Then, the line and error associated with this current triplet are provided as the outputs D_{opt} and $E_{S_{opt}}$ of the algorithm (along with a flag: see below), and the algorithm ends.

For some 'pathological' data sets S, rounding errors occurring in computations on a finite-precision computer may be such that for each triplet, at least one point P_l does not meet $E_{PL}[P_l, D(P_i, P_j, P_k)] \leqslant E_T(P_i, P_j, P_k)$. Then, the algorithm ends after having considered all triplets, but without providing the solution. Because of this possible situation, the algorithm always returns a flag, set to *Solved* or *Not_solved* depending on whether the algorithm has found the solution.

The complexity of this algorithm ranges from n (when the first triplet provides the solution) to n^4 (when all triplets must be tested). It is almost-insensitive to truncations. It provides an almost-exact solution.

```
Algorithm 1
{
For i = 0 to (n − 3)
    For j = (i + 1) to (n − 2)
        For k = (j + 1) to (n − 1)
            {
            Find D(Pi, Pj, Pk) and ET(Pi, Pj, Pk)

            For l = 0 to (n − 1), with l ≠ i, j, k
                If EPL[Pl, D(Pi, Pj, Pk)] > ET(Pi, Pj, Pk)
                    Then next k

            Dopt = D(Pi, Pj, Pk)
            ESopt = ET(Pi, Pj, Pk)
            Flag = Solved
            End
            }

Flag = Not_solved
End
}
```

Figure 2. Weighted minimax linear approximation: first algorithm based on triplets of input data points.

2.5. *Second approach based on triplets of input data points*

Theorem 5: *If the input data set S contains $n \geqslant 3$ points, and if $E_{T_{max}}$ is the maximum among the errors associated with all the triplets of (different) points that may be derived from S, then triplets of points whose associated line D and error E_T are respectively equal to the optimum line D_{opt} and error E_{Sopt}, are all the triplets such that $E_T = E_{T_{max}}$.*

Theorem 5 shows that the problem considered in this section may be solved by successively considering all the triplets of points (with $X_i < X_j < X_k$) that may be derived from S, by computing the line D and error E_T associated with each triplet, and by keeping the highest error E_T thus obtained and the corresponding line D. This error and this line are then the optimum ones. The expression of this algorithm is relatively similar to Fig. 2.

The complexity of this algorithm does not depend on data and is of order n^3. This algorithm is completely insensitive to truncations, and provides an almost-exact solution.

2.6. *Approach based on couples of input data points*

Definition 6: If $\{P_i, P_l, P_k\}$ is a triplet of points of the input data set S, with $X_i < X_l < X_k$, then the corresponding term F_{ilk} is defined as

$$F_{ilk} = \frac{W_l}{X_k - X_i}\left[\frac{X_l - X_i}{W_k} + \frac{X_k - X_l}{W_i}\right] \tag{7}$$

□

Theorem 6: *If the input data set S contains $n \geqslant 3$ points, and if $\{P_j, P_k\}$ is a couple of points of S, with $X_i < X_k$, then*

(i) *Let P_j be an arbitrary point of S such that $X_i < X_j < X_k$ and let D_j and E_{T_j} respectively be the line and error associated with the triplet $\{P_i, P_j, P_k\}$. Let $\varepsilon_j = \pm 1$ define the positions of the points $\{P_i, P_j, P_k\}$ with respect to D_j: $E_{T_j} = \varepsilon_j e_{PL}(P_i, D_j) = -\varepsilon_j e_{PL}(P_j, D_j) = \varepsilon_j e_{PL}(P_k, D_j)$. ε_j may also be defined as follows: $\varepsilon_j = \operatorname{sgn}[e_{PL}(P_i, D_j)] = -\operatorname{sgn}[e_{PL}(P_j, D_j)] = \operatorname{sgn}[e_{PL}(P_k, D_j)]$, where* sgn *is the sign function, defined as:* $\operatorname{sgn}(x) = 1$ *if* $x \geqslant 0$, *and* $\operatorname{sgn}(x) = -1$ *otherwise.*

(ii) *Let S′ be an arbitrary subset of S, composed only of points P_l such that $X_i < X_l < X_k$, and containing at least one point. For each point P_l of S′, let D_l be the line associated with the triplet $\{P_i, P_l, P_k\}$.*
In these conditions, the following equivalence holds

$$[\forall P_l \in S', E_{PL}(P_l, D_j) \leqslant E_{T_j}] \Leftrightarrow \begin{cases} G_{max} \leqslant E_{T_j} \\ \text{and} \\ \text{if } S' \text{ is such that } G_{min} \text{ is assigned: } G_{min} \geqslant E_{T_j} \\ (\text{otherwise this condition disappears}) \\ \text{and} \\ \forall P_l \in S', P_l/F_{ilk} = 1, \varepsilon_j e_{PL}(P_i, D_l) \geqslant 0 \end{cases}$$

where the following notations are used

(1) G_{max} *is the maximum among all the following values:*
the values $\varepsilon_j e_{PL}(P_i, D_l)$ corresponding to all the points of S′,
the values:

$$\frac{F_{ilk}+1}{F_{ilk}-1}\varepsilon_j e_{\mathrm{PL}}(P_i, D_l) \tag{8}$$

corresponding to all the points of S' such that $F_{ilk}<1$, if any.

(2) *$G_{\min}$ is the minimum among all the values*

$$\frac{F_{ilk}+1}{F_{ilk}-1}\varepsilon_j e_{\mathrm{PL}}(P_i, D_l) \tag{9}$$

corresponding to all the points of S' such that $F_{ilk}>1$, if any (if such points exist, $G_{\min}$ is assigned; otherwise, it is not).

An algorithm providing the solution to be considered problem may be deduced from Theorem 6 (see Fig. 3). As in Algorithm 1, this algorithm aims at determining a triplet $\{P_i, P_j, P_k\}$, with $X_i<X_j<X_k$, which meets (6), where D is the line associated with this triplet. However, here (6) is split in two conditions, by separately considering the points situated 'between' P_i and P_k from the point of view of their X coordinates and those situated 'outside'. Equation (6) then reads

$$(a)\quad \forall P_l \in S, X_l \in]X_i, X_k[, E_{\mathrm{PL}}(P_l, D) \leqslant E_{\mathrm{T}}(P_i, P_j, P_k) \tag{10}$$

$$(b)\quad \forall P_l \in S, X_l \notin [X_i, X_k], E_{\mathrm{PL}}(P_l, D) \leqslant E_{\mathrm{T}}(P_i, P_j, P_k) \tag{11}$$

Equation (10) also performs the test on E_{PL} for P_j, but this is not a problem because this test is met by construction for this point. Moreover, the current algorithm consists of successively considering several couples of points $\{P_i, P_k\}$, with $X_i<X_k$. From these couples, the algorithm derives triplets $\{P_i, P_j, P_k\}$ which meet (10) (this is explained below). Each such triplet is then used in the same way as in Algorithm 1, except that only (11) remains to be checked instead of (6). The overall structure of the current algorithm (see Procedure_1 in Fig. 3) is therefore similar to Algorithm 1 (in particular, it returns a flag, Flag_Procedure_1, stating if it has found the solution). Its specific features are to loop on couples (again in 'any' order), to check (11) for each triplet, and above all to build triplets from couples according to a method which will now be explained.

At this stage, a couple of points $\{P_i, P_k\}$, such that $X_i<X_k$ and that there exists at least one point P_l with $X_i<X_l<X_k$, are supposed to have been selected. The goal is then to find if there exists (at least) one point P_j, such that $X_i<X_j<X_k$ and that (10) is met. If there exists such a point (all of them are equivalent as shown below), the resulting triplet is used as explained above. Otherwise, no suitable triplets can be built from the current couple, and another couple is considered (see the first 'Then next k' in Fig. 3(*a*)). As a first step, assume that only the points P_j providing a predefined value for ε_j defined in Theorem 6 are to be accepted. Then, Theorem 6 allows us to replace (10) by an equivalent set of conditions, where S' of Theorem 6 here consists of all the points P_l of S such that $X_i<X_l<X_k$. Moreover, the latter set of conditions allows us to determine P_j as follows. A first loop on all the points of S' is performed, in order to determine $G_{\max}$ and $G_{\min}$ (if it is assigned) corresponding to this ε_j, and to test if the condition $\forall P_l \in S'$, $P_l/F_{ilk}=1$, $\varepsilon_j e_{\mathrm{PL}}(P_i, D_l) \geqslant 0$ is met. If the latter condition is not met, no points P_j are suitable and another couple should be considered (for this predefined ε_j). Otherwise, a second loop on all the points of S' is performed in order to determine if a point P_j of S' is such that $G_{\max} \leqslant E_{\mathrm{T}_j}$ and

$G_{min} \geqslant E_{T_j}$ (this condition is only tested if G_{min} was assigned above) and such that the value ε_j corresponding to this point (as defined in Theorem 6) is equal to the predefined value that it is assumed to have here. If such a point exists, it provides a suitable triplet which is then used as explained above (and there is no use looking for other points P_j for the current couple, because all the resulting triplets are associated with the same line and error). Otherwise, no points P_j are suitable and another couple should be considered (for this predefined ε_j).

ε_j was assumed to be predefined above. In fact, it can take two values, i.e. ± 1. Therefore, the overall search of a point P_j for a given couple $\{P_i, P_k\}$ consists of first considering, for example, $\varepsilon_j = 1$, and then looking for a corresponding point P_j as explained above. If no such point is found, then $\varepsilon_j = -1$ is used in the same way. This approach is implemented in Algorithm 2: Procedure_2 performs the overall search of P_j, while Procedure_3 corresponds to the case when ε_j is already set. Both procedures return a flag stating whether they have found a suitable point P_j. When they find it, they return the line and error associated with the corresponding triplet $\{P_i, P_j, P_k\}$.

The main advantage of this algorithm is that its complexity for each couple is only of order n because it only peforms 1 to 5 loops on part of the points of S. Therefore, the complexity of the overall algorithm ranges from n (when the first couple provides the solution) to n^3 (when all couples must be tested) instead of n^4 for Algorithm 1.

In the preliminary description provided above, the determination of an intermediate point P_j uses the tests $G_{max} \leqslant E_{T_j}$ and $G_{min} \geqslant E_{T_j}$. From a mathematical point of view, strict equality occurs in these tests (e.g. $G_{max} = E_{T_j}$) for any data set when a suitable triplet is used. Therefore, because of rounding errors on finite-precision computers these tests often fail, so that the overall preliminary algorithm often fails. This is avoided by adding a 'safety margin' to these tests (see Fig. 3(*b*), e.g. $G_{max} \leqslant (1+\alpha)E_{T_j}$, with $0 < \alpha \ll 1$). The final algorithm thus obtained is almost insensitive to truncation effects, but only provides an approximated solution, with a precision fixed by α.

2.7. *Combined approaches*

The preferred final algorithms consist of associating the basic algorithms presented above in order to combine their advantages. These combined algorithms contain two steps. If an approximated solution is acceptable, the algorithm that should be called in the first step is the algorithm based on couples of input data points, because this is, on average, the fastest proposed algorithm. Otherwise, i.e. if an almost-exact solution is required, the algorithm that should be called in the first step is the first algorithm based on triplets of input data points, which is somewhat slower. For almost all data sets, this first step provides the solution and the combined algorithm ends at this stage. Otherwise, a second step is performed by calling the second algorithm based on triplets of input data points. This algorithm is much slower, but guarantees that the solution is always found.

3. Minimax decomposition of piecewise functions, without interpolation; application to weighted minimax possibly-discontinuous PWL approximation

3.1. *General problem statement*

Lower and higher levels. In this section, a 'lower level' (LL) is assumed to be available. It is created by defining three main items, i.e. by selecting a basic type of approximating subfunction f_u; by defining the error corresponding to an input data

Algorithm 2

```
Procedure_1 /* Overall algorithm */
{
For i = 0 to (n − 3)
     For k = (i + 2) to (n − 1)
          {
          Call Procedure_2
          If Flag_Procedure_2 = Pj_not_found
               Then next k

          For l = 0 to (i − 1) and l = (k + 1) to (n − 1)
               If E_PL[P_l, D(P_i, P_j, P_k)] > E_T(P_i, P_j, P_k)
                    Then next k

          D_opt = D(P_i, P_j, P_k)
          E_Sopt = E_T(P_i, P_j, P_k)
          Flag_Procedure_1 = Solved
          End
          }

Flag_Procedure_1 = Not_solved
End
}


Procedure_2 /* Search P_j */
{
Call Procedure_3 with ε_j = 1
If Flag_Procedure_3 = Pj_found
     Then {
          Flag_Procedure_2 = Pj_found
          Return
          }
Else {
     Call Procedure_3 with ε_j = −1
     If Flag_Procedure_3 = Pj_found
          Then {
               Flag_Procedure_2 = Pj_found
               Return
               }
     Else {
          Flag_Procedure_2 = Pj_not_found
          Return
          }
     }
}
```

3 (*a*)

```
Procedure_3 /* Search P_j with predefined ε_j */
{
For l = (i + 1) to (k − 1)
     {
     Find D_l associated with {P_i, P_l, P_k}, F_ilk and e_PL(P_i, D_l)

     If (F_ilk = 1) and (ε_j.e_PL(P_i, D_l) < 0)
          Then {
               Flag_Procedure_3 = Pj_not_found
               Return
               }

     If (l = i + 1) or (ε_j.e_PL(P_i, D_l) > G_max)
          Then G_max = ε_j.e_PL(P_i, D_l)

     If (F_ilk < 1) and ([F_ilk + 1]/[F_ilk − 1].ε_j.e_PL(P_i, D_l) > G_max)
          Then G_max = [F_ilk + 1]/[F_ilk − 1].ε_j.e_PL(P_i, D_l)

     If (F_ilk > 1) and ( ( G_min not yet assigned in this call of Procedure_3 )
     or
                                   ([F_ilk + 1]/[F_ilk − 1].ε_j.e_PL(P_i, D_l) < G_min))
          Then G_min = [F_ilk + 1]/[F_ilk − 1].ε_j.e_PL(P_i, D_l)
     }

For l = (i + 1) to (k − 1)
     {
     Find D_l and E_Tl associated with {P_i, P_l, P_k}, and e_PL(P_i, D_l)

     If e_PL(P_i, D_l) ≥ 0
          Then ε_l = 1
     Else ε_l = −1

     If (ε_l = ε_j) and (G_max ≤ (1 + α).E_Tl) and
                    ( ( G_min not yet assigned in this call of Procedure_3 ) or
                                                  ((1 + α).G_min ≥ E_Tl))
          Then {
               Flag_Procedure_3 = Pj_found
               Return ( D_l, E_Tl)
               }
     }

Flag_Procedure_3 = Pj_not_found
Return
}
```

(*b*)

Figure 3. Weighted minimax linear approximation: algorithm based on couples of input data points.

set and to such a subfunction f_u; and by creating algorithms which minimize this error with respect to f_u. This section applies to any such LL and especially to the one developed in §2, which corresponds to the final target of this paper. This section aims at building a 'higher level' (HL) on top of this LL. This consists of defining the same type of items as for the LL, except that this section applies to more complex approximating functions.

Input data. Here again, the input consists of a set S^T (where 'T' stands for Total set, as opposed to the subsets of S^T defined below). This set contains n points P_i, with $i=0$ to $(n-1)$. The point coordinates depend on the considered LL. Only the X_i coordinates are 'seen' in the HL, and they should strictly increase with i (as in §2).

Class of approximating functions. Here, this class contains any function f^T (where 'T' again stands for Total) defined as follows: its domain of definition consists of all real values and is split in m non-empty intervals, where m is a parameter of f^T. These intervals are indexed by the interval index u ranging from 0 to $(m-1)$, and they are denoted I_u. The limits of these intervals are denoted x_u, with x_u strictly increasing with u, i.e. $I_0=]-\infty, x_1[$ for $u=1$ to $(m-2)$ $I_u=[x_u, x_{u+1}[$, and $I_{m-1}=[x_{m-1}, +\infty[$. f^T is split according to the intervals I_u, i.e. it contains m parts and on each interval, I_u, it is equal to a subfunction f_u which belongs to the class of functions corresponding to the LL (for simplicity, the same LL is used on all intervals). When using §2 as the LL, each function f_u is defined as: $f_u(x)=p_u x+q_u$. Then, f^T is a possibly-discontinuous PWL function.

Subsets S_u of input data points. For any u, with $u=0$ to $m-1$, the set S_u is the subset of the set S^T composed of the input data points P_i whose X_i coordinates are inside I_u. When using §2 as the LL, each subset S_u corresponds to the set S of §2 (but the notations n of §2 and of the current section should not be mixed: n of §2 corresponds to the number of points of S_u, which is only part of the total number of points n of S^T considered in this section).

Definition 7—Point/total-function error E_{PFT}: The error between an input data point P_i and an approximating function f^T is defined as

$$E_{PFT}(P_i, f^T)=E_{PFU}(P_i, f_u) \tag{12}$$

where f_u is the subfunction of f^T corresponding to the interval I_u which contains X_i. The definition of the error E_{PFU} between a point and a subfunction is assumed to be available from the LL. When using §2 as the LL, $E_{PFU}=E_{PL}$ as defined in §2. □

Definition 8—Set/total-function error E_{SFT}: The error between an input data set S^T of points P_i, with $i=0$ to $(n-1)$, and an approximating function f^T is defined as

$$E_{SFT}(S^T, f^T)=\max_{i\in\{0\ldots(n-1)\}} [E_{PFT}(P_i, f^T)] \tag{13}$$

□

As explained in §1, this definition corresponds to a worst-case approach. In such an approach, it is natural to define the overall error E_{SFT} of f^T from its local errors for all points P_i. As shown hereafter, E_{SFT} may also be defined from the overall errors of the subfunctions f_u, which are available form the LL. This is better suited to practical algorithms and yields an easier generalization in §4.

Definition 9—Subset/subfunction error E_{SFU}: For any u, with $u=0$ to $(m-1)$, the error between the subset S_u of a set S^T comprising points P_i and the subfunction f_u of an approximating function f^T is defined as

$$E_{SFU}(S_u,f_u)=\max_{P_i\in S_u}[E_{PFU}(P_i,f_u)] \tag{14}$$

□

When using §2 as the LL, $E_{SFU}=E_{SL}$ as defined in §2 (computed only with the points of S^T situated in S_u).

Theorem 7: *The error between a set S^T and an approximating function f^T may also be expressed as*

$$E_{SFT}(S^T,f^T)=\max_{u\in\{0\ldots(m-1)\}}[E_{SFU}(S_u,f_u)] \tag{15}$$

Definition 10—Optimum approximating functions and optimum error: Given an input data set S^T and a number of parts m, the optimum approximating functions are all the m-part approximating functions f^T_{opt}, of the above-defined class, which minimize the error $E_{SFT}(S^T,f^T)$. The optimum error $E_{ST_{opt}}(S^T,m)$ is the error $E_{SFT}(S^T,f^T)$ common to all optimum approximating functions. □

Goal of the investigation, approximation criterion. Given an input data set S^T and a number of parts m, this section aims at determining some corresponding optimum approximating function. The associated minimum error is also to be determined. As explained in §1, this corresponds to a minimax approach. In particular, when using §2 as the LL, this section performs a weighted minimax possibly-discontinuous PWL approximation. Moreover, as explained in §4, this approximation criterion does not take into account the interpolation capability of the approximating functions (while the approach of §4 does).

3.2. *Quantization of the limits of the intervals I_u*

The parameters of f^T to be optimized consist of all interval limits x_u and all parameters of all subfunctions f_u. The basic principle of the algorithms proposed to solve this problem consists of optimizing the parameters x_u by successively considering different 'basic cases', where each basic case corresponds to using a fixed set $\{x_1,\ldots x_{m-1}\}$. In each such basic case, all subsets S_u are fixed, and therefore, all subfunctions f_u are optimized independently by using the algorithms available from the LL. Moreover, for any interval index u and any point P_i, all the values of x_u belonging to $]X_{i-1},X_i]$ yield the same decomposition of S^T in subsets S_u and therefore the same error $E_{SFT}(S^T,f^T)$. So, when looking for the minimum of this error, there is no need to vary all x_u continuously. Instead, the minimum error is found by using only quantized values x_u, which yield an overall set of basic cases defined as follows: for each possible decomposition of S^T into subsets S_u consider a single set $\{x_1,\ldots x_{m-1}\}$ which provides this decomposition. In this paper, the following single set $\{x_1,\ldots x_{m-1}\}$ is considered: as stated above, the equivalent values for each x_u are those belonging to $]X_{i-1},X_i]$. Therefore, the single value which is used is X_i. To sum up, the overall finite set of basic cases considered is defined by two conditions i.e. all x_u are equal to X_i coordinates of points of S^T (with

x_u increasing strictly with u) and these basic cases provide all possible decompositions of S^T into subsets S_u. With this approach, each x_u is completely defined by the index l_u of the corresponding point of S^T: for $u=1$ to $m-1$, $x_u=X_{l_u}$. These parameters l_u are used instead of x_u below. In addition, the conventions $l_0=0$ and $l_m=n$ are used. Each basic case is then defined by the set $\{l_1,\dots l_{m-1}\}$. The last step of this approach consists of defining all the values of this set $\{l_1,\dots l_{m-1}\}$ which should be considered, i.e. which correspond to a possible decomposition of S^T into subsets S_u. This requires a preliminary definition.

Definition 11: The minimum *width* of any subset S_u of S^T is the minimum number of input data points P_i that it is allowed to contain. It is denoted W. □

W depends on the LL. When using §2 as the LL, $W=2$. This results from the fact that decompositions of S^T, yielding only two points in a subset S_u, should be considered because they are needed to guarantee that $E_{SFU}(S_u,f_u)=0$ is achieved. Conversely, going down to one point is useless because it does not decrease the minimum of this error $E_{SFU}(S_u,f_u)$ and it can only degrade the errors on the other intervals (because they contain more points than in the previous case), so that it degrades the resulting error E_{SFT}.

When this limit W is introduced, S^T should contain enough points to have at least W points available per part u of f^T, i.e. the problem is relevant only when: $n>Wm$. Moreover, each subset S_u consists of the points of S^T whose indices range from l_u included to l_{u+1} excluded. Therefore, the possible sets $\{l_1,\dots l_{m-1}\}$ are all those such that

$$\forall u, u=0 \text{ to } (m-1), \quad l_{u+1}\geq l_u+W \tag{16}$$

The possible sets $\{l_1,\dots l_{m-1}\}$ resulting from (16) may be defined by using an approach where all the possible values of l_1 are considered, and for each such value all the possible values of l_2 are considered, and for each such couple of values all the possible values of l_3 are considered ... and so on, until all l_u are defined. Then, it is easily shown that the range required for each l_u to consider all possible basic cases is

$$\forall u, \quad u=1 \text{ to } (m-1), \quad l_u=l_{u-1}+W \text{ to } n-(m-u)W \tag{17}$$

3.3. *Approach based on a complete loop on l_u*

An algorithm for solving the problem considered in this seciton may be directly derived from the above discussion. Its basic version consists of performing an outermost loop on all the possible values of l_1 (defined by (17)). For each such l_1, a loop on all the corresponding possible values of l_2 (defined by (17)) is performed ... and so on, until the innermost loop on l_{m-1}. For each such complete set $\{l_1,\dots l_{m-1}\}$, all subfunctions f_u are optimized independently, as stated above, and the corresponding optimum errors E_{SFU} are obtained. This yields a current locally optimum function (i.e. a function optimized with respect to all f_u for these fixed l_u). This function is denoted f^T (see Fig. 4). Its error E_{SFT} is derived from all E_{SFU} according to (15). The overall algorithm consists of keeping a partially optimum function f^T_{opt} (and its error $E_{ST_{opt}}$), which is one of the functions providing the lowest error E_{SFT} among all the locally optimum functions f^T considered up to the current basic case. This partial optimum is possibly updated for each basic case. When all

Algorithm 3

```
Procedure_1 /* Overall algorithm */
{
Assign parameter m of f^T
Assign parameter l_0 of f^T: l_0 = 0
Assign parameter l_m of f^T: l_m = n

Flag_opt = 0 /* states that E_STopt has not yet been assigned */

Call Procedure_2 ( 1, 0 )
}


Procedure_2 ( u, E_left) /* Recursive procedure */
{
If u < m
     Then for l_u (of f^T) = l_{u-1} (of f^T) + W to n - (m - u).W
               {
               Call Procedure_3
               Call Procedure_2 ( u + 1, E_leftmid )
               }

Else {
     Call Procedure_3

     If (E_leftmid < E_STopt) or (Flag_opt = 0)
          Then {
               f^T_opt = f^T
               E_STopt = E_leftmid
               Flag_opt = 1
               }
     }
}


Procedure_3 /* Perform optimization on I_{u-1} */
{
Optimize f_{u-1} (of f^T)
Find corresponding E_SFUopt (i.e. optimum E_SFU(S_{u-1}, f_{u-1}))

If u = 1
     Then E_leftmid = E_SFUopt
Else E_leftmid = max[E_left, E_SFUopt]
}
```

Figure 4. Minimax decomposition of piecewise functions (without interpolation) with application to weighted minimax possibly-discontinuous PWL approximation: approach based on a complete loop on l_u.

possible basic cases have been considered, it is a globally optimum function and the corresponding error $E_{ST_{opt}}$ is the optimum error.

This algorithm may then be improved in various ways. First, when considering a new set $\{l_1, \ldots l_{m-1}\}$, some of its elements l_u (having the lowest indices) have the same value as in the previous set. Therefore, there is no use re-optimizing the corresponding subfunctions f_u of the current locally optimum function. Instead, they are kept from one set $\{l_1, \ldots l_{m-1}\}$ to the next one. To this end, they are forwarded from each loop level u (which varies l_u) to the next one, along with f_{u-1} which is the only subfunction optimized by the loop level u. The corresponding errors E_{SFU} are also forwarded to the next loop level. More precisely, only their maximum needs to be forwarded, since only E_{SFT} defined by (15) should be available at the end. This maximum among errors E_{SFU} may be defined as follows, with respect to Definition 12 provided below. From the point of view of the calling loop level which sends this maximum E_{SFU} to the next loop level, this value is equal to the error $E_{leftmid}$ of this calling level. From the point of view of the called level which receives it, it is its error E_{left}.

Definition 12: When an input set S^T is processed according to the loops on all l_u defined above, and when the considered loop level is u (i.e. l_u is varied), the set of intervals I_0 to I_{m-1} associated with the currently considered approximating function f^T is split into three parts with respect to this level u.

(1) The left part consists of the intervals I_0 to I_{u-2} (if any). These are the intervals which are not varied in the current level, because l_1 to l_{u-1} are fixed (by the previous levels) in this level.

(2) The middle part consists of the interval I_{u-1}. This interval is varied in the current level, because l_u is varied by this level. However this interval is fixed for each such l_u (and therefore for each call to the next levels).

(3) The right part consists of the intervals I_u to I_{m-1} (if any). These are the intervals which are varied in the current level, even for a fixed l_u, because l_{u+1} to l_{m-1} (if any) are varied in the calls to the next levels that are 'included in the current level.

For any such part of intevals or any union of such parts, an associated error is defined as the maximum of the errors E_{SFU} corresponding to the intervals I_u which comprise this part or union of parts. In particular, the following errors are used in this paper

$$\forall u,\ u=2 \text{ to } m,\ E_{left}(S^T, f^T, u) = \max_{v \in \{0 \ldots (u-2)\}} [E_{SFU}(S_v, f_v)] \tag{18}$$

$$\forall u,\ u=1 \text{ to } m,\ E_{leftmid}(S^T, f^T, u) = \max_{v \in \{0 \ldots (u-1)\}} [E_{SFU}(S_v, f_v)] \tag{19}$$

$$\forall u,\ u=1 \text{ to } m,\ E_{mid}(S^T, f^T, u) = E_{SFU}(S_{u-1}, f_{u-1}) \tag{20}$$

$$\forall u,\ u=0 \text{ to } m-1,\ E_{right}(S^T, f^T, u) = \max_{v \in \{u \ldots (m-1)\}} [E_{SFU}(S_v, f_v)] \tag{21}$$

□

The proposed algorithm may also be improved as follows. If a software implementation of this algorithm explicitly contained the loops on l_u mentioned above, the number of loops written in the source code would fix the value m. To allow

the user to select m on each run, a recursive program is preferred. All these considerations yield Algorithm 3 (Fig. 4), where the recursive procedure Procedure_2 receives the following parameters: the index u of the parameter l_u that is varied in this recursion level, the error E_{left} corresponding to this current *called* level as explained above, and other common variables such as the partly assigned current function f^T.

3.4. *Specific problem statement*

The remainder of this section only concerns the case when the LL has two properties. It should first be such that whatever u and S_u, if the number of input data points contained by S_u is equal to W, then $E_{\text{SFU}_{\text{opt}}}(S_u) = L$, where $E_{\text{SFU}_{\text{opt}}}$ corresponds to the optimization performed by the LL with respect to f_u for a given S_u, and where the limit L on $E_{\text{SFU}_{\text{opt}}}$ is a constant value, corresponding to the considered LL. The LL should also be such that: $\forall u, \forall S_u, \forall P_i, \forall f_u$, $E_{\text{SFU}}(S'_u, f_u) \geqslant E_{\text{SFU}}(S_u, f_u)$, where S'_u is the union of S_u and P_i. It is easily shown that these properties are met when using § 2 as the LL (with $L = 0$). The goal of the investigation is the same as in the beginning of this section, but here the properties of the LL allow us to determine that some sets $\{l_1 \ldots l_{m-1}\}$ cannot provide optimum functions (or that these functions are also provided by other sets). This allows us to create approximation algorithms which skip such sets $\{l_1 \ldots l_{m-1}\}$ and which are thus much faster than the algorithm presented above (which of course also applies here).

3.5. *Approaches based on partial loop on l_u*

Consider a given input data set S^T, an approximating function f^T to be optimized and a given loop level u. For each value of l_u, two errors are defined as follows. The optimum error $E_{\text{mid-opt}}$ associated with the 'middle part' is obtained by optimizing E_{mid} (see Definition 12) with respect to f_{u-1}. The optimum error $E_{\text{right-opt}}$ associated with the 'right part' is obtained by optimizing E_{right} (see Definition 12) with respect to l_{u+1} to l_{m-1}, and to all parameters of all sub-functions f_u to f_{m-1}.

For each loop level u, Algorithm 3 described above progressively increments l_u from its minimum to its maximum possible value. Thus, thanks to the properties met by the LL here, $E_{\text{mid-opt}}$ increases or remains constant and $E_{\text{right-opt}}$ decreases or remains constant. Moreover, when l_u is equal to its minimum possible value, $E_{\text{mid-opt}}$ is equal to L because S_{u-1} contains W points. Similarly, when l_u is equal to its maximum possible value, $E_{\text{right-opt}}$ is equal to L because each of the subsets S_u to S_{m-1} contains W points. All these conditions mean that $E_{\text{mid-opt}}$ 'crosses' $E_{\text{right-opt}}$ for a particular value of l_u. More precisely, in the general case, a single value l_u^0 of l_u meets the following conditions: $\forall l_u, l_u \leqslant l_u^0$, $E_{\text{mid-opt}} < E_{\text{right-opt}}$ and $\forall l_u, l_u > l_u^0$, $E_{\text{mid-opt}} > E_{\text{right-opt}}$. Then the optimum value of l_u (for the current values l_1 to l_{u-1}) is either l_u^0 or $(l_u^0 + 1)$ because when l_u is varied further from these values in any direction, either $E_{\text{mid-opt}}$ or $E_{\text{right-opt}}$ increases, so that their maximum increases and therefore E_{SFT} increases (or remains constant: in this case, other values l_u may also yield the optimum, but only the above two values are needed to determine this optimum). Among these two values, the optimum is the one which minimizes $\max(E_{\text{mid-opt}}, E_{\text{right-opt}})$. This suggests modifying Algorithm 3 as follows: in each level u, start the loop on l_u in the same way as in Algorithm 3 but when the 'crossing' occurs, i.e. when $E_{\text{mid-opt}}$ becomes higher than $E_{\text{right-opt}}$, stop this loop, determine the optimum l_u among the current value and the previous one as explained above (i.e. by

testing $\max(E_{\text{mid-opt}}, E_{\text{right-opt}})$), compare the corrresponding locally optimum function f^{T} (and error E_{SFT}) with the partial optimum one, update the latter if needed, and return to the previous loop level. On rare occasions, strict equality $E_{\text{mid-opt}} = E_{\text{right-opt}}$ may occur for one (or several) l_u^0. Then this l_u^0 (or any of them) is the optimum l_u and is used in the same way as above.

A second algorithm based on a partial loop on l_u consists of also stopping the loop on l_u if both $E_{\text{mid-opt}}$ and $E_{\text{right-opt}}$ become lower than E_{left}, because improving $E_{\text{mid-opt}}$ and $E_{\text{right-opt}}$ further will not decrease E_{SFT} below E_{left} anyway. This yields faster operation and does minimize E_{SFT} (i.e. the maximum among all E_{SFU}), which is the only requirement in this section. However, this approach only meets this requirement, while the first version (and Algorithm 3) in fact minimizes all E_{SFU}, thus taking full advantage of the allocated number of parts m.

3.6. *Approaches based on a dichtomy on l_u*

The underlying principle of the first approach, based on a partial loop on l_u presented above, is to consider the difference $(E_{\text{mid-opt}} - E_{\text{right-opt}})$. This is a monotonic function of l_u, which starts from a value lower than or equal to zero for the minimum possible value of l_u, and which ends with a value higher than or equal to zero for the maximum possible value of l_u. The algorithm then consists of determining the 'zero' of this function, or in fact the adjacent values of l_u for which the sign of this function changes, since l_u and therefore this function only has discrete values. In the algorithm presented above, this zero search was performed by progressively increasing l_u. Another approach consists of performing a classical dichotomy on l_u, starting with an initial dichotomy interval corresponding to the minimum and maximum possible values of l_u, and ending when the interval is composed of two adjacent values, which contain the optimum in the same way as above (i.e. test $\max(E_{\text{mid-opt}}, E_{\text{right-opt}})$). Here again, a faster version is obtained by taking into account E_{left} in order to end the loop on l_u prematurely when possible, and only E_{SFT} is thus optimized instead of all E_{SFU}. These dichotomic algorithms are, on average, much faster than the previous ones, since the number of values of l_u that they test at each recursion level only varies as the logarithm of the number of possible values of l_u. They are therefore the preferred algorithms.

4. Minimax decomposition of piecewise functions, with interpolation. Possibly discontinuous PWL application

If the approximating function f^{T} is to be used only for X coordinates equal to coordinates X_i of points P_i of the specified set S^{T}, the optimization algorithms presented in §3 should be used. Conversely, if it is to be used for any X, these algorithms yield desired behaviour for any X except for the sub-interval $I'_u =]X_{(l_{u+1}-1)}, X_{l_{u+1}}[$ of each interval I_u, with $u=0$ to $m-2$. This may be explained as follows, for such a fixed index u, As stated in §3, each subfunction f_u is optimized by taking into account the points whose indices range from l_u included to l_{u+1} *excluded.* Then, for any X situated in I'_u, f^{T} is equal to f_u which is used in the extrapolation mode, i.e. this X is not surrounded by points P_i taken into account in the optimization of this f_u (all these points are 'on the left' of X). This may yield a low approximation accuracy. This problem is solved by also using the point $P_{l_{u+1}}$ in the optimization of f_u: this point is situated 'on the right' of the considered X, so that f_u is thus only used in the interpolation mode.

This approach yields the same general and specific problem statements as in § 3, except for the following items. For $u=0$ to $(m-2)$, the set S_u contains all the input data points that it contained in § 3, but also the point which limits I_u on the right. By using the quantized values x_u of § 3 and the corresponding variables l_u, S_u consists of the input data points P_{l_u} included to $P_{l_{u+1}}$ *included*. For $u=(m-1)$, the set S_u is defined in the same way as in § 3 because $P_{l_{u+1}}$ does not exist. The error E_{SFT} is here only defined according to (15) (with respect to the new subsets S_u).

The problems thus defined are solved by using the same approach as in § 3, except that the subsets S_u do not contain the same points. This yields the following modifications. First, the problem is relevant only when $n \geqslant (W-1)m+1$. Then, the possible sets $\{l_1 \ldots l_{m-1}\}$ are all those defined by the following conditions (again with $l_0=0$ and $l_m=n$): $\forall u, u=0$ to $(m-2)$, $l_{u+1} \geqslant l_u+W-1$ and for $u=(m-1)$, $l_{u+1} \geqslant l_u + W$. By using the same loop structure as in § 3, these conditions yield the following possible range of values for each l_u

$$\forall u, u=1 \text{ to } (m-1),\ l_u=l_{u-1}+W-1 \text{ to } n-1-(m-u)(W-1) \tag{22}$$

This yields the same algorithms as in § 3, except that the subsets S_u defined in the current section are used and that the range of values for each l_u is defined by (22).

5. Weighted minimax continuous PWL approximation

The problem treated here is the same a in § 4, except that only the LL of § 2 is used (the approach may be easily extended to another LL) and that the optimum is searched only among continuous PWL functions. This problem is defined with respect to § 4 rather than § 3 because it is more natural to take the interpolation capability into account here, since smooth continuous functions are considered. The proposed algorithms provide functions which may be slightly sub-optimum. This is not a problem because this section is only a step towards the next section which 'restores' optimality.

In the basic stage of this section, an m-segment possibly-discontinuous PWL function f^T is available. A continuous PWL function g^T is then derived as follows. g^T also has m segments. It is defined by $(m+1)$ points, i.e. the $(m-1)$ limits between its segments, and two other points respectively situated in its leftmost and rightmost segments. These last two points are respectively made equal to P_0 and P_{n-1}. The other $(m-1)$ points have the same X coordinates as the corresponding limits between adjacent segments of f^T, i.e. X_{l_u}. To assign the Y coordinate of any of these points, two 'hints' are available, i.e. the left and right limits of f^T at this X (i.e. for $X=X_{l_u}$). These limits are respectively equal to the values of the susbfunctions corresponding to the two adjacent segments, i.e. $f_{u-1}(X_{l_u})$ and $f_u(X_{l_u})$. The Y coordinate of the point of g^T is made equal to one of these limits, i.e. to the one corresponding to the segment where the higher error $E_{SFU}(S_v, f_v)$ occurs. This method is coherent with minimax approaches, which only take into account the items which yield the highest errors. However, g^T thus obtained is not guaranteed to be optimum.

A first algorithm for solving the overall problem of this section consists of calling one of the algorithms of § 4, which provides a function f^T, and then deriving g^T as explained above. g^T is the output of this algorithm. An alternative is an algorithm similar to one of those of § 4 except that, for each set $\{l_1 \ldots l_{m-1}\}$, after determining the current function f^T, the corresponding continuous function g^T is derived as

explained above. The error E_{SFT} corresponding to S^T and g^T is then computed and compared with the partial (sub-)optimum among the continuous functions obtained up to now, and these (sub-)optimum functions and errors are updated if needed. At the end of the algorithm, this yields the selected (sub-)optimum continuous function. This approach is slower than the first but likely to provide a function g^T closer to the optimum, because it selects the (sub-)optimum set $\{l_1 \dots l_{m-1}\}$ according to the errors corresponding to g^T, not to f^T.

6. Weighted minimax continuous PWL approximation with quantized parameter values

The problem considered here is the same as in § 5, except that the X and Y coordinates of the points which define an approximating function are only allowed to have integral values. A simple solution to this problem consists of performing an exhaustive search on these discrete values in a given domain. However, this leads to a combinatorial explosion for the numbers of segments m used in real applications. Instead, approaches similar to those of § 5 are proposed here.

More precisely, in a basic stage, it is assumed that a continuous PWL function g^T is available and that the X coordinates of its points are integers. The latter condition is achieved by using a data set S^T such that all X_i are integers. Then, a continuous PWL function h^T defined by points having integral X and Y coordinates is derived by only modifying the Y coordinates of the points of g^T as follows. The simplest method only consists of rounding the Y coordinates of the points of g^T to the nearest integers. In a more elaborate version, for the Y coordinate of each point of h^T, a small set of integral values surrounding the corresponding Y coordinate of g^T is successively considered. The error E_{SFT} associated with each such function h^T is determined, and the function providing the lowest error is kept. This version is slower than the simplest method but is likely to provide, most often, the optimum function within the type of functions considered here, because only a discrete set of functions exists and the optimum function h^T is likely to be close to g^T. This version thus 'erases' the sub-optimality of the algorithms of § 5.

Two approaches, similar to those of § 5, are then proposed to solve the overall problem of this section. In the first, one of the algorithms of § 4 is first called, thus providing a function f^T. Then, a continuous function g^T is derived as explained in § 5. Finally, the procedure described above in this section derives the corresponding function h^T, which is the output of the algorithm. Conversely, in the second approach the determination of f^T, g^T, h^T is performed for each set $\{l_1 \dots l_{m-1}\}$, and the partial (sub-)optimum function and error which are progressively updated correspond to an continuous PWL function with integral coordinates. Here again, the second approach is slower but likely to provide a function closer to the optimum (especially when based on the algorithms of § 4 which use a larger number of sets $\{l \dots l_{m-1}\}$).

7. Conclusions and applications

In this paper, we have mainly preesented algorithms for a weighted minimax PWL approximation, which led to various cases, depending on the exact type of PWL appproximating functions used. This may be summarized as follows. Two major cases were first presented, which respectively consist of using single-line approximating functions or possibly-discontinuous PWL ones (with interpolation

capability taken into account or not). The algorithms proposed for these cases provide optimum functions. The other two cases consist of requiring the PWL approximating functions to be continuous and/or defined by points which have integral X and Y coordinates. These two specific cases were studied because they correspond to practical applications, as shown at the end of this section. The algorithms proposed for these two cases provide functions which are not guaranteed to be optimum, but which are likely to be most often very close to the optimum. Future work in this domain might consist of looking for optimum algorithms.

All these algorithms obviously apply to simple (i.e. unweighted) minimax PWL approximation: all the weights W_i should then be set to 1. However, restricting *a priori* the investigation to unweighted minimax PWL approximation allows us to derive improved algorithms. In particular, the algorithm of § 2, based on couples of input data points, may then be simplified. Also, algorithms providing a worst-case complexity which is only a small fraction of n^2 (instead of the complete n^3 in this paper) may then be created (they are not presented in this paper where the target application uses weights, as explained by Deville 1994 a).

We have also presented algorithms which have wider applicability than the weighted minimax PWL approximation. This remark mainly concerns the algorithms of § 3 and 4, which optimize the domains of operation of the subfunctions of any type of piecewise functions according to a (possibly weighted) minimax criterion, by taking interpolation capability into account or not. To a lower extent, this also concerns the basic stages of §§ 5 and 6.

These optimization algorithms were investigated because they were needed in an industrial application, i.e. automatic TV tuner alignment. This application (and others) is presented in a companion paper (see Deville 1994 a) and the remainder of this section aims only at showing how it leads to the approximation problems considered in this paper. This application requires us to build a transcoder which receives a digital word (defining the desired frequency of operation of the TV tuner), and which provides a corresponding adequate digital word (defining the control voltage provided to a filter of the tuner). This transcoder should have a very simple hardware structure, in order to be easily embedded into one of the tuner chips. This yields the following constraints on the form of its transfer function. First, in order to simplify the structure of its logic circuitry, which computes the value of this function for the current input value during TV operation, this function is only allowed to involve linear computations and comparisons. The transfer function of the transcoder is thus restricted to possibly-discontinuous PWL functions. Then, one should minimize the size of the transcoder memory required to achieve a given performance (defined by the error E_{SFT} as explained below). Therefore, only *continuous* PWL approximating functions are accepted: when used to approximate relatively smooth functions, for a given number of segments m, they yield a slightly higher optimum error than possibly-discontinuous PWL functions, but they significantly reduce the number of parameters needed to define a function. As a result, they most often achieve a given performance with a lower memory size. Finally, the logic circuitry of the transcoder is restricted to operate only with integers, in order to simplify its structure. Therefore, only approximating functions defined by points having integral X and Y coordinates are accepted. The functional form of § 6 is thus obtained.

Among this type of transfer function, a particular function is selected as follows. Each manufactured tuner should have a specific transcoder transfer function, because of component spreads between tuners. This function is therefore derived

from measurements which are performed for each manufactured tuner and which yield the above-mentioned points P_i defining the ideal behaviour of the tuner. The transcoder transfer function should be chosen so as to fit this set of points in 'the best way'. This requires us to define a figure for measuring the performance of a tuner (defined by a set of points P_i) and of a given transcoder transfer function. This performance figure is fixed by the considered application and is equal to the maximum frequency mistuning over the complete band of operation (see Deville 1994 a). As shown by Deville (1994 a), this performance figure is estimated by an overall error which may be rapidly defined as the maximum among weighted errors associated with the points P_i (where the weights are defined by the application and derived from the coordinates of the points P_i and possibly from additional information). Therefore, the transfer function f^T of the transcoder is chosen so as to minimize this overall error, thus yielding the weighted minimax approximation criterion considered in this paper.

References

Batruni, R., 1991, A multilayer neural network with piecewise-linear structure and back-propagation learning. *IEEE Transactions on Neural Networks*, **2**, 395–403.

Box, G. E. P., Hunter, W. G., and Stuart Hunter, J., 1978, *Statistics for Experimenters* (New York: Wiley).

Deville, Y., 1994 a, Optimum piecewise-linear transcoders—Part 2: Application to automatic TV tuner alignment and to electronic tuning. *International Journal of Electronics*, **77**, 845–862; 1994 b, Weighted minimax piecewise-linear approximation and minimax decomposition of piecewise functions. Report from the Laboratoires d'Electronique Philips.

Kahlert, C., and Chua, L. O., 1990, A generalized canonical piecewise-linear representation. *IEEE Transactions on Circuits and Systems*, **37**, 373–383; 1992, The complete canonical piecewise-linear representation—Part I: The geometry of the domain space. *IEEE Transactions on Circuits and Systems—I*, **39**, 222–236.

Kevenaar, T. A. M., and Leenaerts, D. M. W., 1992, A comparison of piecewise-linear model descriptions. *IEEE Transactions on Circuits and Systems—I*, **39**, 996–1004.

Lin, J., and Unbehauen, R., 1992, Canonical piecewise-linear approximations. *IEEE Transactions on Circuits and Systems—I*, **39**, 697–699.

Papoulis, A., 1984, *Probability, Random Variables, and Stochastic Processes* (Singapore: McGraw-Hill).

Yamamura, K., 1992, On piecewise-linear approximation of nonlinear mappings containing Gummel-Poon models or Shichman-Hodges model. *IEEE Transactions on Circuits and Systems—I*, **39**, 694–697.