Some Applications of Interval Analysis to Statistical Problems

Vincent Vigneron

Some Applications of Interval Analysis to Statistical Problems

Vincent Vigneron

[1] IBISC CNRS FRE 2873
Université d'Evry
91020 Evry Cedex, France
{vincent.vigneron}@ibisc.univ-evry.fr
[2] CES CNRS-UMR 8174 Equipe SAMOS-MATISSE
75634 Paris cedex 13,France
vigneron@univ-paris1.fr

Abstract. This paper contribution is about *guaranteed* numerical methods based on interval analysis for approximating sets, and about the application of these methods to vast classes of statistical problems. 'Guaranteed' means here the inner and outer approximations of the sets of interest are obtained, which can be made as precise as desired, at the cost of increasing the computational effort. It thus becomes possible to archieve tasks still thought by many to be out of the reach of numerical methods, such as finding all solutions of sets of non-linear equations and inequalities or a global optimizer of possible multi-modal criteria.

Notations

Throughout the text, the following conventions are used: lower case letters in italics such as x or y_i denote scalar variables and elements of vectors. Vectors are printed in bold. A row vector is denoted by the transpose operator, *i.e.* $\mathbf{x}^T$. Uppercase bold characters denote matrices, for instance $\mathbf{X}$.

1 Introduction

Interval computation is a special case of computation on sets, and set theory provides the formulations for interval analysis [1,2]. Set and interval mathematics come from the same general theory developped during the 30's by the french School of Topology: if a number a and a bound b of a approximate the value of some number x such that $|x - a| \leq b$, then *interval mathematics* tells us that x is in the interval $[a - b, a + b]$. Hence, the type of an interval is dual: at the same time *number* and *set*, with evident implications in set arithmetics [3]. For instance, suppose a real axis provided with an order relation $\leq$, the interval $[\mathbf{x}]$, bounded by $\underline{x}$ and $\overline{x}$, is a closed connected subset of real numbers $\{x|\underline{x} \leq x \leq \overline{x}\}$. Interval arithmetic follows from order properties and basic operations on real numbers or vectors extend in a natural way to intervals. However, the arithmetical rules for intervals differ from those for real numbers. For instance, $x^2 + x + 100 = (x + \frac{1}{2})^2 + \frac{399}{4}$ whereas $[x]^2 + [x] + 100$ differs from $([x] + \frac{1}{2})^2 + \frac{399}{4}$, as illustrated in the following example. For instance, at $[x] = [0, 1]$, $[x]^2+[x]+100 = [0, 1]^2+[0, 1]+100 = [100, 102]$, and $([0, 1]+\frac{1}{2})+\frac{399}{4} = [\frac{1597}{16}, \frac{1621}{16}]$. The first result is a pessimistic approximation of the image set of $x^2 + x + 100$ at $[0, 1]$ whereas the second (and beteter one) is equal to this image set.
A box or *vector* of intervals $[\mathbf{x}]$ is the cartesian product of n intervals and is noted $[x_1] \times \ldots \times [x_n]$, with $[x_i] = [\underline{x_i}, \overline{x_i}], i = 1, \ldots, n$. Notions introduced for intervals

extend without difficulty to boxes. The set of all boxes of $\mathbb{R}^n$ is denoted $\mathbb{IR}^n$. Note that $]-\infty,\infty[\times\ldots\times]-\infty,\infty[$ is an element of $\mathbb{IR}^n$. Consider the situation where we have a model which acts as a function f, mapping (inputs) x to (outputs) y. This model f might be quite complex, with multiple input parameters and with different kinds of uncertainty represented on them: information available on inputs may be rich or sparse, so-called "aleatory" and may be made known through objective measurements. Mathematically inputs might be represented by probability or possibility distributions, by strong or sparse collections of data points, by simple intervals, or even by non-quantified linguistic expressions.
Given f, how can we propagate the uncertainty on x to y through f? Moreover, how can we do so in a way which respects all the original uncertainty quantifications as provided, making no unnecessary assumptions? How can we do such in a way which uses *only*, but *all of* what we are given?
In this paper, we propose an approach for solving such problems: we assume that experimental points (both inputs and outputs) are modeled as intervals and provide exact solutions. The contribution of this article is two fold: firstly, it is shown that interval analysis can directly be applied to perform optimization, yielding to close form expressions of results. Secondly, , short illustrative examples including a nonlinear process and a blind source separation problem are given which show the significative improvement of the approach in comparison with standard linear identification.

2 A refresher on parameter estimation

Consider some function $f:\mathcal{X}\subset\mathbb{R}^n\rightarrow\mathcal{Y}\subset\mathbb{R}^s$, where $\mathcal{Y}$ is a bounded set and suppose we wish to construct a model $g:\mathbb{X}\subset\mathcal{X}\rightarrow\mathbb{Y}\subset\mathcal{Y}$, where $\mathbb{X}$ and $\mathbb{Y}$ are some domains of interest, by choosing a parameter vector $\mathbf{p}\in\mathbb{R}^p$ so that, mathematically speaking,

$$\mathbf{y}=f(\mathbf{x})=g(\mathbf{x},\mathbf{p})+e(\mathbf{p}), \tag{1}$$

for all $\mathbf{x}\in\mathbb{X}$ and $\mathbf{y}\in\mathbb{Y}$, where the error in approximation, $e(\mathbf{p})$, is as small as possible.
We suppose that all that is available to choose the parameters $\mathbf{p}$ in g is some part of the *unknown* function f in the form of the input-output data pair associations. The ith input-output pair for the system f is denoted by $(\mathbf{x}_i,\mathbf{y}_i)$, where $\mathbf{x}_i\in\mathbb{X},\mathbf{y}_i\in\mathbb{Y}$ and $\mathbf{y}_i=f(\mathbf{x}_i)$. This may correspond for instance to $n+s$ scalar measurements corresponding to various experimental conditions on a static process or on a dynamical one. The row vector $\mathbf{z}_i=(\mathbf{x}_i^T\mathbf{y}_i^T)\in\mathbb{R}^{n+s}$ denotes one particular data sample. Stacking N consecutive samples on top of each other gives the data matrix

$$\mathbf{Z}=\begin{bmatrix}\mathbf{z}_1\\ \mathbf{z}_2\\ \vdots\\ \mathbf{z}_N\end{bmatrix}\in\mathbb{X}\times\mathbb{Y}. \tag{2}$$

The purpose of parameter estimation is, for instance, to find $\mathbf{p}$ such that $g(\mathbf{x},\mathbf{p})$ best fits $\mathbf{y}$ in a sense to be specified. In [4], the parameters are considered admissible if the error $e(\mathbf{p})$ belongs to some prior compact set of admissible error $\mathbb{E}\subset\mathbb{R}^s$. For instance, $\mathbb{E}$ may be the box defined as

$$\mathbb{E}=\{e|e^-\leq e\leq e^+\}, \tag{3}$$

where e^+ and e^- are some prior bounds. One is then interested in finding the set $\mathbb{S}$ of all values of $\mathbf{p}$ such that the error is admissible, *i.e.* $\mathbb{S}=\{\mathbf{p}|e(\mathbf{p})\in\mathbb{E}\}$. This set has

been called *membership set, likelihood set* and *posterior feasible set*. If the data were generated by a statistical model $g(\mathbf{x}, \mathbf{p}^*)$, where $\mathbf{p}^*$ is some true value of the parameters and if $e(\mathbf{p}^*) \in \mathbb{E}$, then $\mathbb{S}$ contains $\mathbf{p}^*$. Thus, $\mathbb{S}$ provides an accurate description of the uncertainty with which $\mathbf{p}^*$ is estimated [4].
If the reciprocal of g exists and is denoted g^{-1}, $\mathbb{S}$ is defined as $\mathbb{S} = g^{-1}(\mathbf{y} - \mathbb{E}) = g^{-1}(\mathbb{Y})$, where $\mathbb{Y} = \mathbf{y} - \mathbb{E}$ is the *measurement set*. In other words, for any $\mathbf{p} \in \mathbb{S}$ there exists $e \in \mathbb{E}$ such that $\mathbf{y} = g(\mathbf{x}, \mathbf{p}) + e$.
System identification (*i.e.* function approximation) amounts to adjusting $\mathbf{p}$ using information from $\mathbf{Z}$ so that $g(\mathbf{x}, \mathbf{p}) \approx f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{X}$. Measured and model outputs never match perfectly in practice, but differ as $e(\mathbf{p})$. An obvious modeling goal must be that this discrepancy is "small" in some sense that is archieved by the value of the *approximation error* we wish to bound. Such a bound is for example

$$\sup_{\mathbf{x} \in \mathbb{X}} \|f(\mathbf{x}) - g(\mathbf{x}, \mathbf{p})\|, \tag{4}$$

which requires that f is known everywhere. The problem is that we only know the part of f given by $\mathbf{Z}$, and it is the only evaluation we can make based on known information. Let us consider a system with imprecise input and output x and y resp., which are readings from unreliable sensor ("noisy data"). To simplify exposition, we shall only consider output errors :

$$e_i(\mathbf{p}) = y_i - g(\mathbf{x}_i; \mathbf{p}),\ i = 1, \ldots, s \tag{5}$$

but other types of errors could be considered as well. We assume that these errors should satisfy $\underline{e_i} \leq e_i(\mathbf{p}) \leq \overline{e_i},\ i = 1, \ldots, s$ to be "admissible", where $\underline{e_i}$ and $\overline{e_i}$ are known lower and upper prior bounds of the approximation error resulting from technical specifications or pointing out how far we can go in accepting discrepancies between our data and model outputs. Note that $e_i(\mathbf{p})$ is an interval.
Let $\mathbf{y}$ be the vector of all data $y_i, i = 1, \ldots, s$ collected on a given system, and let $\mathbf{g}(\mathbf{x}; \mathbf{p})$ be the vector of all corresponding model outputs $g(\mathbf{x}_i; \mathbf{p}),\ i = 1, \ldots, s$.
Equation (5) can then be rewritten as

$$\mathbf{e}(\mathbf{p}) = \mathbf{y} - g(\mathbf{x}; \mathbf{p}), \tag{6}$$

Picked in intervals, equation (6) gives:

$$[\mathbf{e}(\mathbf{p})] = [\mathbf{y}] - g([\mathbf{x}]; [\mathbf{p}]), \tag{7}$$

The model (7) is said to be *admissible* if $\mathbf{p}$ is such that $\mathbf{e} \in \mathbb{E}$, *i.e.* errors should satisfy

$$[\mathbf{y}] - [g]([\mathbf{x}]; [\mathbf{p}]) \in [\mathbf{e}]([\mathbf{p}]). \tag{8}$$

$[g]$ is an *inclusion function* of g that returns an *enveloping box* guaranteed to contain the image by g of any given box $[\mathbf{x}]$ included in the domain of g. $[\mathbf{g}]([\mathbf{x}])$ is a box such that $\mathbf{g}([\mathbf{x}]) \subset [\mathbf{g}]([\mathbf{x}])$. The including function $[\mathbf{g}]$ is easy to compute for usual elementary functions that can be obtained by composition of elementary operations such as $+, -, \times, /, \exp, \tan, \sin, \ldots$ by replacing each of these elementary operations by their inclusion function in the formal expression of g.
When no efficient algorithm exists for its computation, $[g]$ can be approximated by an inclusion function $\mathbb{G} : \mathbb{R}^n \to \mathbb{R}^p$ satisfying $\mathbf{g}([\mathbf{x}]) \subset [\mathbf{g}]([\mathbf{x}])$ and such that:

$$w([\mathbf{x}]) \to 0 \Rightarrow w([\mathbb{G}]([\mathbf{x}])) \to 0. \tag{9}$$

where $w([\mathbf{x}])$ is the width of the box $[\mathbf{x}]$, defined as the length of its largest side(s).

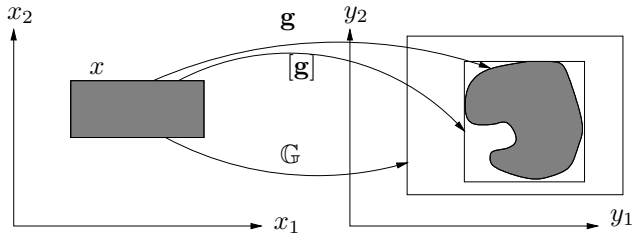


Fig. 1. Minimal inclusion function $[g]$ and inclusion function $\mathbb{G}$ of a function g.

Figure 1 illustrates conditions $\mathbf{g}([\mathbf{x}]) \subset [\mathbf{g}]([\mathbf{x}])$ and (9).
Let $\widehat{\mathbb{S}} = \{\mathbf{p} | g([\mathbf{x}]; [\mathbf{p}]) \in \mathbb{Y}\}$. Then

$$g([\mathbf{x}]; [\mathbf{p}]) \in \mathbb{Y} \Leftrightarrow \mathbf{p} \in g^{-1}(\mathbb{Y}) \quad \text{and} \quad \mathbf{p} \in [\mathbf{p}](0) \tag{10}$$

$$\Leftrightarrow \quad \mathbf{p} \in [\mathbf{p}](0) \cap g^{-1}(\mathbb{Y}) \tag{11}$$

where $[\mathbf{p}](0)$ is the search domain. Thus

$$\widehat{\mathbb{S}} = [\mathbf{p}](0) \cap g^{-1}(\mathbb{Y}), \tag{12}$$

and characterizing $\widehat{\mathbb{S}}$ is a *set inversion problem*.
We shall say that $[\mathbf{p}]$ is *feasible* if $[\mathbf{p}] \subset \mathbb{S}$, *unfeasible* if $[\mathbf{p}] \cap \mathbb{S} = \varnothing$, else $[\mathbf{p}]$ is *ambiguous*.
To perform computation of $\mathbb{S}$ in an approximate but guaranteed way, an interval analysis algorithm is applied to compute the possible interval range $[\mathbf{p}]$ of $\mathbf{p}$. The interval computation stage is detailed in the following.

3 Identification by set characterization

Methods allowing to implement interval analysis are relatively few and date from the nineties and among them, one may quote the Moore's algorithm [3] and SIVIA proposed by Jaulin [5]. Most of the methods for estimating parameters are based on computations performed at point values of the parameter vector. The main interest in the notion of paving is to replace point values by subsets of the parameter space. For simplicity, we will use pavings based upon boxes.
A *paving* of a compact subset $\{\mathbb{P}\} \subset \mathbb{R}^n$ is a set of non overlapping boxes with nonzero width such that the union of these boxes corresponds to $\{\mathbb{P}\}$. A subpaving $\mathbb{K}$ of $\mathbb{P}$ is a subset of $\mathbb{P}$. Upon completion, the algorithm encloses $\mathbb{S}$ between two compact sets corresponding to 2 subpavings (Fig. 3). Let $\mathbb{E}$ be the feasible error set. Initialisation is performed by setting $\mathbb{Y} = \mathbf{y} - \mathbb{E}$. The principle is as follows:

a). Define an initial box of interest $[\mathbf{p}](0)$ within which the search will be performed
b). Compute a paving $\{\mathbb{P}\}$ of $[\mathbf{p}](0)$
c). Compute the image $g([\mathbf{x}]; [\mathbf{p}])$ for each box of this paving. Three situations must then be considered (see figure 2).

$$\begin{cases} [g]([\mathbf{x}]; [\mathbf{p}]) \subset [\mathbb{Y}] \Rightarrow [\mathbf{p}] \subset \mathbb{S} \quad \text{so that} [\mathbf{p}] \text{is } \textit{feasible} \\ [g]([\mathbf{x}]; [\mathbf{p}]) \cap [\mathbb{Y}] = \varnothing \Rightarrow [\mathbf{p}] \cap \mathbb{S} = \varnothing \quad \text{so that} [\mathbf{p}] \\ \text{is } \textit{unfeasible} \\ \text{otherwise, } [p] \text{ is } \textit{indetermined}. \end{cases}$$

The exploration algorithm performs a recursive implementation of the principle that has just been described: a *bisection algorithm* splits each box of the subpaving into smaller boxes whenever needed until the width of the box becomes smaller than some tolerance parameter ϵ to be specified by the user. Cutting is carried out again as long as the boxes contain solutions or stops if the boxes do not contain any.

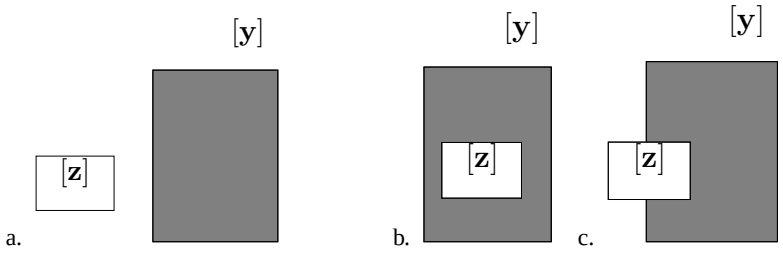


Fig. 2. Feasibility of boxes: a test function allows to distinguish the cases (a-c) represented in this figure. Let $[\mathbf{z}]$ and $[\mathbf{y}]$ be two boxes. Suppose that $[\mathbf{n}] = [\mathbf{z}] \cap [\mathbf{y}]$. a) $[\mathbf{n}] = \varnothing$ means that $[\mathbf{z}]$ has an empty intersection with $[\mathbf{y}]$. b) $[\mathbf{n}] = [\mathbf{z}]$, so that $[\mathbf{z}]$ is included inside $[\mathbf{y}]$. c) $[\mathbf{n}] \subset [\mathbf{z}]$ and $[\mathbf{n}] \neq [\mathbf{z}]$. $[\mathbf{z}]$ intersects $[\mathbf{y}]$, so that we cannot conclude. The box $[\mathbf{z}]$ can be split again.

We shall split $\mathbb{P}$ iteratively into three subpavings $\underline{\mathbb{S}}$, $\mathbb{S}$ and $\overline{\mathbb{S}}$ corresponding to the sets of all feasible, unfeasible and indetermined boxes respectively, as plotted in figure 3.

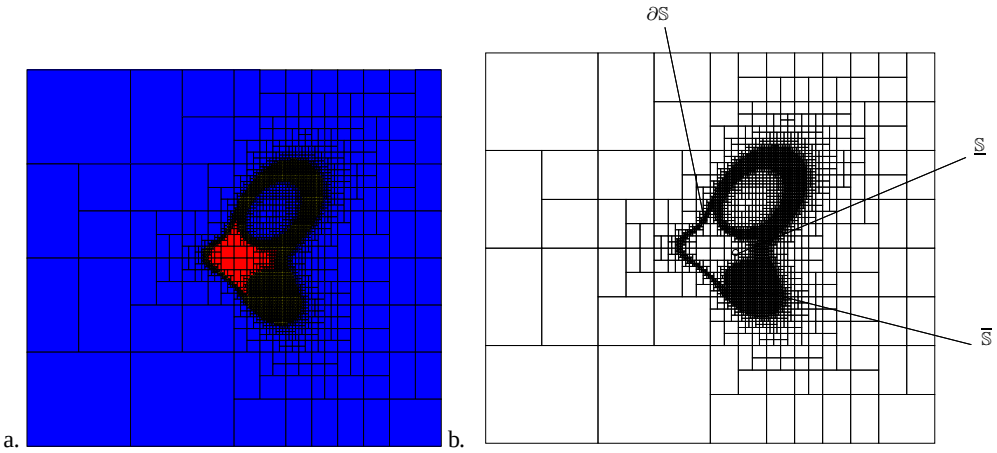


Fig. 3. a. Regular paving of a box: accepted, rejected and indetermined subpavings are respectively coloured in red, blue and yellow. b. $\{\underline{\mathbb{S}}$ and $\overline{\mathbb{S}}\}$ brackets the portion of $\mathbb{S}$ contained in $[\mathbf{p}](\mathbf{0})$.

These subpavings satisfy the following relations :

1. $\{\underline{\mathbb{S}}\} \subset \mathbb{S} \subset \{\underline{\mathbb{S}}\} \cup \{\overline{\mathbb{S}}\}$
2. $\text{vol}(\{\underline{\mathbb{S}}\}) \leq \text{vol}(\mathbb{S}) \leq \text{vol}(\{\underline{\mathbb{S}}\}) + \text{vol}(\{\overline{\mathbb{S}}\})$

3. $[\{\underline{\mathbb{S}}\}] \subset [\mathbb{S}] \subset [\{\underline{\mathbb{S}}\}] \cup [\{\overline{\mathbb{S}}\}]$

Provided that $\mathbb{S}$ is full, this means that the pair $\{\underline{\mathbb{S}}; \overline{\mathbb{S}}\}$ defines a neighbourhood $\partial\mathbb{S} \triangleq \overline{\mathbb{S}} \backslash \underline{\mathbb{S}}$ of $\mathbb{S}$ with a diameter that can be chosen arbitrarily small.
The previous algorithm makes an extensive use of a *stack* L of boxes, *i.e.* a dynamical structure on which only 3 operations are possible: at any time, one may put an element on top of the list, remove the top element or test the stack for emptiness. We define the required accuracy ϵ for the paving $\mathbb{P}$ as the maximum widh that an indetermined box can have. In the following, the principal plane of a box is the symmetry plane of this box that is orthogonal to the axis $i \in \{j | w([\mathbf{p}]) = w([p_j])\}$, where the operator "width" $w([\cdot])$ of a box is the length of its largest side.
Let $[\mathbf{p}](0)$ be the box considered at iteration k. Initialisation is performed by setting $k = 0, \mathsf{L} = \varnothing, \underline{\mathbb{S}} = \overline{\mathbb{S}} = \varnothing$. The recursive algorithm can be described as follows:

Algorithm BISECT

INPUTS
data: $\mathbf{y}$
inclusion function: $[g]([\cdot])$
feasible error set: $\mathbb{E}$
prior feasible box: $[\mathbf{p}](0)$
accuracy for the paving: ϵ
INITIALIZATION
$\mathbb{Y} = \mathbf{y} - \mathbb{E}$;
stack: $L = \varnothing$
iteration: $k = 0$
$[\mathbf{p}] = [\mathbf{p}](0)$;
ITERATION k
Begin
step 1: if $[g]([\mathbf{x}]; [\mathbf{p}](k)) \subset [\mathbb{Y}]$ then $\underline{\mathbb{S}} := \underline{\mathbb{S}} \cup [\mathbf{p}]$ and $\overline{\mathbb{S}} := \overline{\mathbb{S}} \cup [\mathbf{p}]$;
step 2: else if $[g]([\mathbf{x}]; [\mathbf{p}](k)) \cap [\mathbb{Y}] = \varnothing$, then $\underline{\mathbb{S}} := \underline{\mathbb{S}} \cup [\mathbf{p}]$; then unstack $[\mathbf{p}](k)$ as unfeasible;
step 3: else if $w([\mathbf{p}](k)) \leq \epsilon$, then $\overline{\mathbb{S}} := \overline{\mathbb{S}} \cup [\mathbf{p}](k)$;
else cut $[\mathbf{p}](k)$ along the principal plane and stack the resulting boxes in L.
step 4: if the stack is not empty, then unstack and store the resulting box in $[\mathbf{p}](k+1)$;
$k = k + 1$; go to step 1;
End

Table 1. Recursive implementation of the bisection algorithm.

The union of all the boxes in the list L returned by the program contains $\mathbb{S}$; the partition $\mathbb{P}$ consisting of feasible, unfeasible and indetermined boxes can be plotted in the parameter space in the case the space dimension is less than 4 (see fig. 3).

4 Discussion

The method of *set characterization* introduced in section 3 appeals to some comments. Upon completion, this approach encompasses *all* the acceptable values of the parameter vector in a set that is fully characterized by BISECT: $\underline{\mathbb{S}}$ and $\overline{\mathbb{S}}$ will tend to $\mathbb{S}$ from inside and outside when $\epsilon \rightarrow 0$. Since $\overline{\mathbb{S}}$ is a finite union of boxes guaranteed to contain

the portion of $\mathbb{S}$ of interest, it is very convenient for implementing set-theoretic manipulations [6,4].
The advantages of this approach are threefold:

(i) no assumption is made on the image fonction g,
(ii) no statistical assumption on the modeling error is required,
(iii) any bounded error can be treated independently from its origin (modeling and/or measurement error).

An other advantage of the proposed approach is that the input-output roles of the variables x and y can be reversed since the linking function $\mathbf{g} : x \to y$ can be run *forward* as well as *backward* when using interval analysis. Subpavings form a useful class of objects for manipulating statistical estimations.
The algorithm requires a possibly very large search box $[\mathbf{p}](0)$ to which $\mathbb{S}$ is guaranteed to belong. Solvers split the search box into an union of boxes (the *paving*) with guaranteed error bounds (*i.e.* mathematically valid) [7] (see section 3). The paving is built by the solver itself. A computer program can represent a set of (eventually disjoint) intervals as a *list* L. The precision of the solver is controlled by coefficients specifying, for example, the width ϵ of the smallest boxes of the paving, or the accuracy in the localization of a global optimum. The computing time of the solver can increase quickly with the dimension and size of the list L.
Special care must be taken in avoiding memorizing unnecessary information, otherwise the quantity of memory required to store the paving of $\mathbb{S}$ will increase linearily at each iteration, which may result into a memory overflow even for problems of modest dimension.
One can observe that the parameter space is not isotropic because the sensitivities of g with respect to the various components of $\mathbf{p}$ are not of the same order of magnitude. The basic bisection technique suggested in Tab. 1 may not be efficient enough. The problem is then how to choose the fastest bisection policy that results in a convergence as fast as possible. Jaulin *et al.* [4] suggests the bisections of $[\mathbf{p}]$ into boxes $[\mathbf{p_1}]$ and $[\mathbf{p_2}]$ that minimize $\text{vol}(g([\mathbf{x}];[\mathbf{p_1}]))+\text{vol}(g([\mathbf{x}];[\mathbf{p_2}]))$. If $[\mathbf{p}]$ is not ambiguous, this policy will tend to avoid classifying as indetermined. Experiments seem to show that this can improve the efficiency of the solver when the anisotropy is severe.

5 Application to statistical problems

Example 1 (Parameter estimation). In this example, a simplified version of the problem explored by Jaulin and Walter[3] [4] is given. The vector comprising all The numerical values of the corresponding interval data are given by $\mathbf{y} = ([1.18, 2.00], [0.62, 2.26], [0.40, 2.21], [1.09, 1.27], [0.32, 1.81]$
The set $\widehat{\mathbb{P}}$ to be characterized consists of the set of variable vectors $([p_1], [p_2])^T$ such that the graph of the function :

$$f(\mathbf{p}, t) = p_2 \exp(-p_1 t), \tag{13}$$

crosses all the data bars of Figure 4. In this simulated example, the $[y_i]$ were computed by adding a random error interval with radius $\rho_i = 0.5|y_i| + 1$ to the y_i. The initial box domains for the parameters p_1 and p_2 may be arbitrarily large, for example

$$[p_1] = [-10,000; 10,000] \ \textit{and} \ [p_2] = [-10,000; 10,000], \tag{14}$$

i.e. no prior information is available on the parameters. The feasible set for the parameters is given by (12), where the search domain $[\mathbf{p}](0)$ is $[-1, 5] \times [-5, 5]$.

[3] The extension of the method to multiple-output problems is straightforward.

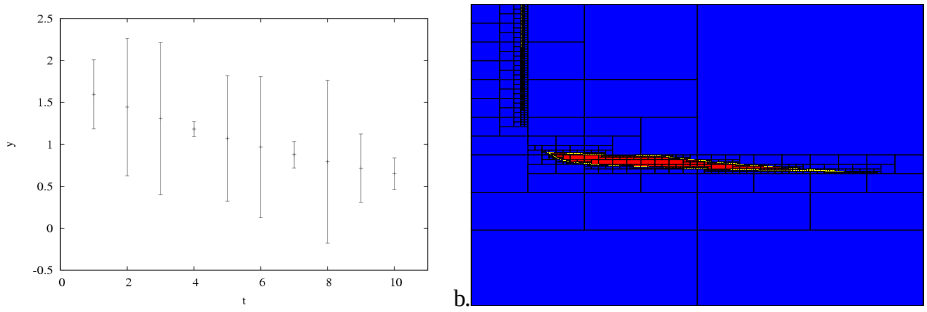


Fig. 4. a) Measurement times and corresponding interval data. b) Top-view of the paving generated by BISECT to bracket the solution set (in red) in the parameter space. The outer frame corresponds to the box $[-1, 5] \times [-5, 5]$.

In less than 1s, on a PENTIUM IV, BISECT generates the pavings of figure 4, thus bracketing the posterior feasible set for $\mathbf{p}_\alpha$ between the inner and the outer approximations. Figures 4.b gives a top-view representation of $\mathbf{p}$, *i.e.* the bounded support of the variables p_1 and p_2 that is consistent with the equation (13) and the domains (14).

Example 2 (Curve estimation). The curvature $\kappa(t)$ of an arbitrary twisted curve $\mathcal{C}$ measures the rate of change of the tangent when moving along the curve. It measures, so to speak, the deviation of the curve from a straight line in the neighbourhood of any of its points. It is quite easy to derive an analytic expression of the curvature which is valid when $\mathcal{C}$ is represented by an allowable parametric representation $\mathbf{x}(t)$:

$$\kappa(t) = \frac{\mathbf{x}' \times \mathbf{x}''}{|\mathbf{x}'|^3}, \tag{15}$$

where $\times$ denotes the vector product. Derivatives with respect to time are denoted by primes, *e.g.* $\mathbf{x}' = \frac{d\mathbf{x}}{dt}$ and $\mathbf{x}'' = \frac{d^2\mathbf{x}}{dt^2}$.

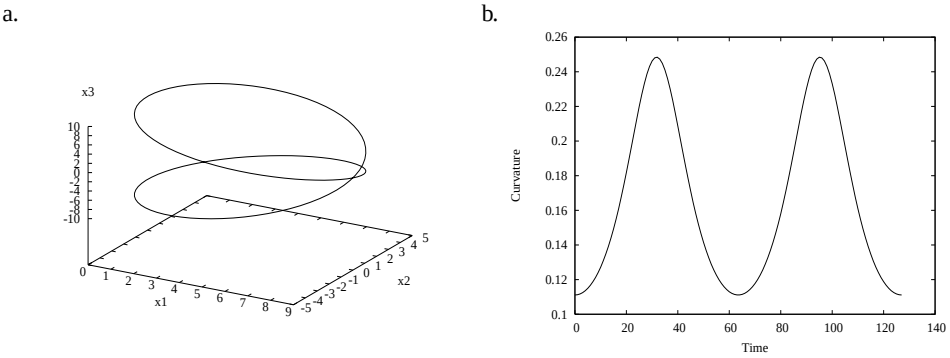


Fig. 5. a. 3-dimensional representation of the Clelia curve. b. Graph of the function $\kappa(t)$.

When $\mathcal{C}$ is a *clelia* curve (see figure 5.a), the cartesian representation[4] is:

$$\mathbf{x}(t) = (R\cos nt\cos t, R\cos nt\sin t, R\sin nt) \tag{16}$$

with t as a parameter. Figure 5.b plots the domain of $\kappa(t)$.
A statistical approach assume that a set of N input-output data pairs $(\mathbf{x}_i, \kappa_i)_{i=1}^N$ is available. Recall that $\mathbf{x}_i \in \mathbb{R}^3$ are vectors and κ_i is scalar. The data set is split into a training and a validation data set. A neural network such as represented in Fig. 6 is designed for supervised learning and prediction task from a particular $\mathbf{x}$. Such architecture is called *multi-layer perceptron* (MLP). Hidden units are placed between the features units and the predictions units. A deterministic MLP trained by a method such as backpropagation [8], can implement any input-output function provided that the number of hidden neurons n_h is sufficiently large.

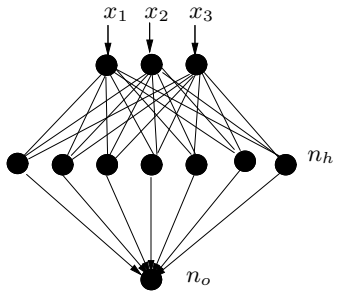
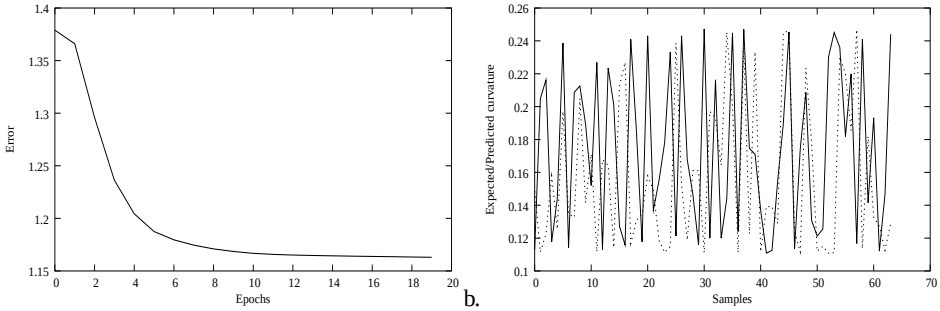


Fig. 6. A deterministic two-layer feedforward neural network with input $\mathbf{x} = (x_1, x_2, x_3)$.

Fig. 7.a depicts the *training error* performed on the training data set.



a. b.

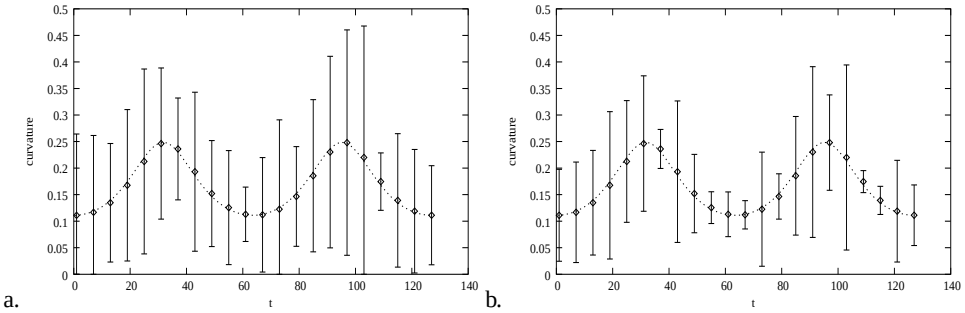
Fig. 7. a. Training error. b. Measured outputs (–) and simulated outputs (. . .) computed by the neural network.

With increased number of hidden neurons, the accuracy of the resulting neural system with respect to the training data set is improved, but the ability of the model to gen-

[4] *Clelia* curve was studied by Guido Grandi in 1728.

eralize for inputs (test set) may be degraded as illustrated in our results (Fig. 7.b). In a similar way, descreasing the output error improves the accuracy with respect to the training data set, but accuracy in the presence of inputs different from the training data set is degraded. These tests illustrate some effects in parameter choice on the resulting neural model.

A second approach consists in using interval analysis tools. In this simulated example, we assume that the values taken by κ are imprecise and sampled at the rate $n\frac{\pi}{11}$: the prior intervals $[\kappa_i]$ are computed by adding a centered error interval to the associated measurement κ_i. The solution set $\widehat{\mathbb{S}}$ to be characterized consists of all the values of $\mathbf{p} = (R, n)^T$ such that the graph of the function $\kappa(t)$ crosses all the data bars of figure 8.a. The dataset is made of 22 data, so very few for a learning model.



a. b.

Fig. 8. a) Experimental data (⧫) together with their uncertainty intervals and graph of the function $\kappa(t)$. b) Posterior feasible intervals for the κ_i superposed on the graph of the function $\kappa(t)$.

For $\epsilon = 0.03$, BISECT generates the subpaving represented in Figure 9 in 7s on a PENTIUM IV. The prior box for the parameters is taken as $[\mathbf{p}] = [-10, 10] \times [-10, 10]$. More interesting, one may want to generate the posterior feasible set for the κ_i (posterior estimates) from the formula as can be seen in figure 8.b.

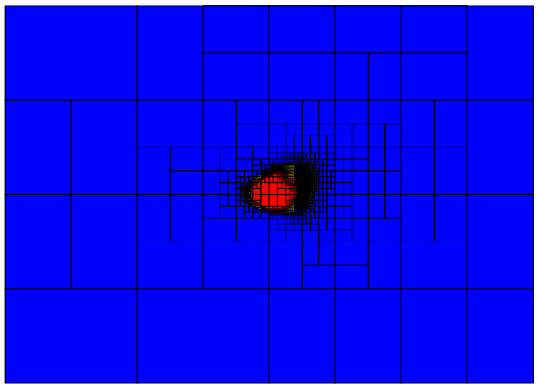


Fig. 9. Top-view of the paving generated by BISECT to bracket the solution set (in red) in the parameter space. The outer frame corresponds to the box $[-10, 10] \times [-10, 10]$.

Example 3 (Blind source separation). The problem adressed here is the recovery of n unknown independent sources $s_i(t)$ from the observation of a n linear mixtures x_i. In matrix and vector notations, this model reads $\boldsymbol{x} = A\boldsymbol{s}$, where $\boldsymbol{s} = (s_1, s_2)^T$ is the vector of sources, $\boldsymbol{x} = (x_1, x_2)^T$ the vector of observations and $A = \{a_{ij}\}$ the mixture matrix. It is a noise-free, time-free model. To recover a vector $\boldsymbol{y}$ close to the source vector $\boldsymbol{s}$ knowing the vector $\boldsymbol{x}$ only, one should estimate some inverse of A, denoted as B. The corresponding estimate of $\boldsymbol{s}$ is $\boldsymbol{y} = B\boldsymbol{x}$. It should be noted, however that the matrix A (or its inverse) is not identifiable from the observations (see e.g.[9]): even if we can extract n independent components, we do not know their ordering. This implies that there exists a *freedom of permutations* of the original signals. The magnitudes of the original signals s_i are also not recoverable, because a scalar mutiple of s_i, cs_i of s_i by a constant c cannot be distinguished from multiplication of the ith column of A by the same constant c. Therefore, therefore,we can recover only a permuted and rescaled version of the sources, *i.e.* we can obtain at best PDA^{-1}, where P is a permutation matrix and D is nonsingular scaling matrix.
The important question is: does the independence of the components of $\boldsymbol{y}$ imply necessarily the *separability* of the mixing model ? The answer to this question is positive in the linear instantaneous domain: the transformation which maps a non-Gaussian random vector with independent components to a random vector with independent components is unique, up to some trivial transformation. This property is a direct result of the Darmois-Skitovich theorem [10]. A solution to the problem (with 2 sources) was first adressed by Herault and Jutten [11] who compute

$$\begin{cases} y_1 &= x_1 + b_{12}y_2 \\ y_2 &= x_2 + b_{21}y_1. \end{cases} \tag{17}$$

where b_{12} and b_{21} are adaptive weights adjusted by means of an adaptation law based on the product of 2 nonlinear functions f and g. The sources are assumed to be zero-mean (*i.e.* $\mathbb{E}[s_1] = \mathbb{E}[s_2] = 0$), stationary and independent.
The independence of the signals means that these sources must have zero covariance, *i.e.* $\mathbb{E}[s_1 s_2] = 0$ and therefore $\mathbb{E}[y_1 y_2] = 0$. These constitute a basis for the adaptation rule considered in [11] to learn the coefficients b_{ij}.

$$\frac{db_{12}}{dt} = \mu y_1^3 y_2, \quad \frac{db_{21}}{dt} = \mu y_2^3 y_1, \tag{18}$$

where μ is a positive constant. It is well-known that the equilibrium points are solutions of $\mathbb{E}[y_1^3 y_2] = 0$ and $\mathbb{E}[y_1 y_2^3] = 0$. However this does not *guarantee* that $\begin{pmatrix} 1 & b_{12} \\ b_{21} & 1 \end{pmatrix}$ converges to the inverse of the mixing equation, even locally.

Let $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ be the (unknown) separating matrix. The solutions of the following system:

$$\begin{pmatrix} 1 & b_{12} \\ b_{21} & 1 \end{pmatrix}^{-1} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{19}$$

reads $b_{12} = \frac{a_{12}}{a_{22}}$ and $b_{22} = \frac{a_{21}}{a_{11}}$, or $b_{12} = \frac{a_{11}}{a_{21}}$ and $b_{22} = \frac{a_{22}}{a_{12}}$. □
Comon *et al.* [12] and Sorouchyari [13] investigate the convergence properties of the algorithm and perform a stability analysis for such a network. They demonstrate that there are *exactly* 4 paired equilibrium points (see figure 10): indeed, if the point (b_{12}^*, b_{21}^*) is a equilibrium point, then the point $(\frac{1}{b_{12}^*}, \frac{1}{b_{21}^*})$ is also a solution (see e.g. [13]).
One can show that such a pair of equilibrium points is on a line passing through the origine of the (b_{12}, b_{21}) plane. But only one of these stationary points will be a *stable*

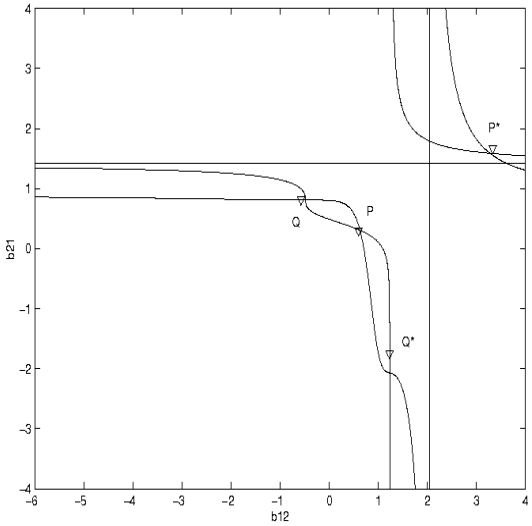


Fig. 10. Theoretical equilibrium points of the system in the (b_{12}, b_{21}) plane.

separating solution [14].

As an illustration, consider the discrete time model in which the data have been generated by simulating for $k = 1, \ldots, 500$: $\begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} = \begin{pmatrix} 1 & 0,6 \\ 0,3 & 1 \end{pmatrix} \begin{pmatrix} s_1(k) \\ s_2(k) \end{pmatrix}$, where $s_1(k) = \sin(7,3kT_e)$, $s_2(k) = \sin(4kT_e)$, $T_e = 0,2$. Prior interval $([x_1(k)], [x_2(k)])$ are obtained by adding a random white noise $n(k) \sim \mathcal{U}_{[-\epsilon,\epsilon]}$, with $\epsilon = 0.01$. Fig. 11.a plots the sources and Fig. 11.b the observed data.

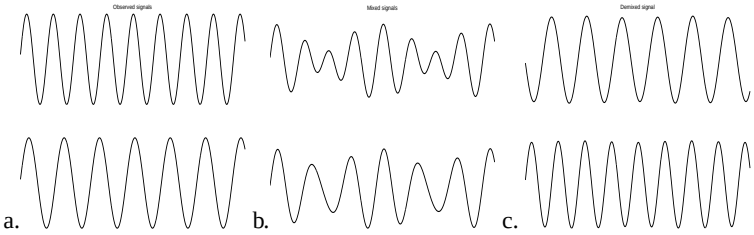


Fig. 11. a) The unknown sources b) the observations c) The separated sources.

An Newton algorithm can be used to find the equilibrium points of the solution. Figure 12.a shows that the algorithm is capable to find the four paired equilibrium points the similarity with Fig. 10 is relevant. The interval algorithm generates also in the (b_{12}, b_{21}) plane (Fig. 12.b and c) the subpaving of the parameters satisfying the independence constraint in less than 1s on a PENTIUM III. After completion, the contracted intervals

in the figures 12.b and c include the true values of the parameters. Once the equilibirum point is identified, it is used for separating the signals (Fig. 11.c).

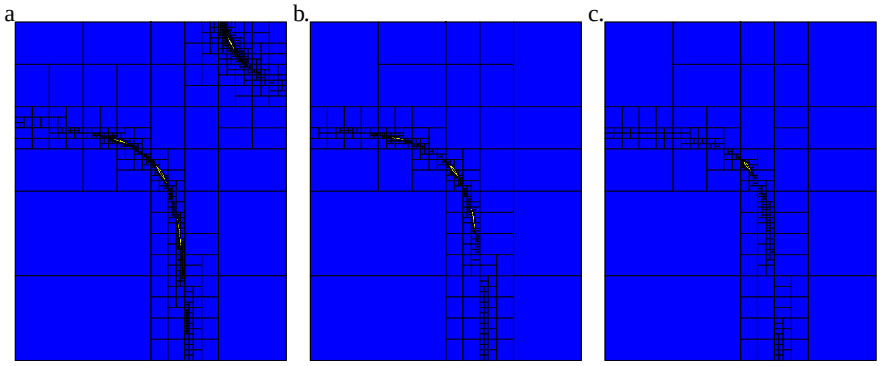


Fig. 12. a) Theoretical equilibrium points of the system b,c) stable equilibrium points obtained by interval analysis in the (b_{12}, b_{21}) plane.

6 C++ implementation

The identification algorithm presented in section 3 was implemented in C++.
A `BOX` class was designed, which allows the programmer to ignore the details of interval implementation. The `BOX` class contains the properties of the intervals, the definition of arithmetical operations on intervals, input-output functions, etc.
The instanciation of a `BOX` is very similar to a `float`, thus one may write

```
int main()
{
  BOX y(2,3);
  BOX x;
// ...
}
```

It is now possible to create and initialize intervals. The instruction `BOX& operator=(const BOX&);` assigns the value of the `BOX` argument to the calling `BOX` instance. For example, a possible implementation of the addition is

```
BOX& operator+(const BOX& a,const BOX& b)
{
  BOX res;
  res.inf=a.inf+b.inf;
  res.sup=a.sup+b.sup;
  return res;
}
```

with evident notations. The implementation of the inclusion functions for the standard mathematical functions obey the usual syntax for mathematical functions (`Cos`, `Sin`, `Exp`,...) and can be used to evaluate properties of an interval, such as its width, or its centre, etc. They are stored in a module `math.cpp` which constitues an interval counterpart to the standard mathematical library.

One may prefer to use a ready-made library: PROFIL/BIAS is a library for interval computation. It runs under many operating systems, from UNIX to DOS. The library can be downloaded from `ftp://ti3sun.tu-harburg.de/pub/profil/`. BIAS (Basic Interval Arithmetic Subroutine), has been written in C in the spirit of the FORTRAN BLAS library (Basic Linear Algebra Subroutines). The basic arithmetic operations, rounding controls and elementary mathematical functions are implemented. For instance, an inclusion function for the exponential function could be implemented as follows:

```
BOX Exp(const BOX& x)
{return BOX(exp(Inf(x)),exp(Sup(x)));}
```

The implementation of algorithms for interval analysis requires specific tools, to be stored in a module `Bisect.cpp`. This module implements, for instance, the bisection of an interval vector `x` accross its `i`th dimension *via* the functions `Lower` and `Upper`, which compute the two interval vectors resulting from the bisection.

```
BOX Lower (const BOX& x, int& i);
BOX Upper (const BOX& x, int& i);
```

`Bisect.h` starts with the definition of a new type of variable `BOX_BOOL` corresponding to interval booleans, *i.e.* variables that can take their values `IB_TRUE, IB_FALSE` and `IB_INDET`. Thus, an interval boolean test can be passed to `Bisect.cpp` as a parameter.
The code `Bisect.cpp` uses a recursive structure (the routine calls itself). It its result is true or false, then a message indicating the result is displayed, followed by a return statement. Else the box is indeterminate, and its width is computed. If it turns out to be lower than the precision parameter `eps`, then a message is displayed, followed by a return statement. Else the current box is bisected using `Lower` and `Upper`, and `Bisect.cpp` is called for each of the resulting subboxes.

```
//----------------------------------------
#include ''Bisect.h''

void Bisect(repere& R,const BOX& P,
                      float eps,float alpha)
{
  if (Width(P)<eps)
  {
    R.DrawBox(P,XFIG_YELLOW);
    //draw a yellow box
    return;
  }
  BOX_BOL test=Inside(P,alpha);
  if (test==IB_TRUE)
    R.DrawBox(P,XFIG_RED); //draw a red box
  else if (test != IB_FALSE)
  {
    R.IncDivision();
    P1=Lower(P,MaxW(P));
    P2=Upper(P,MaxW(P));
    Bisect(R,P1,eps,alpha);
    // Bisect is called recursively
    Bisect(R,P2,eps,alpha);
  }
```

`MaxW()` function returns the index of the first component with maximal width of this box by reference to an `int`. The function `Bisect()` produces a console and a graphic

outputs with the function `R.DrawBox()`. To use `Bisect.cpp`, it now suffices to define the test `Inside()` to be inverted and to write the function `main()`, which will call `Bisect()`:

```
//----------------------------------------
#include ''Bisect.h''

// Definition of the test
BOX_BOL Inside(BOX& P,float alpha)
{
  float y[11];
  BOX_BOL dedans;
  for (i=1;i<=10;i++)
  {
    float e=(1-alpha);
    dedans=dedans&&In(P[1]*Exp(-P[2]*i)
                 ,interval(y[i]-e,y[i]+e));
  }
  return dedans;
};


void Estim (repere& R,BOX& P,float a)
{
  y[1]=1.597038;//data
  ...
  y[10]=0.649307;
  R.DrawBox(P,XFIG_BLUE);
  Bisect(R,P,eps,alpha)
}

void main()
{
  BOX P0(2);   //search box
  P0[1]= INTERVAL(-10,10);
  // initiate search box
  P0[2]= INTERVAL(-10,10);
  for (int k=0;k<=9;k++)
    {
      float a=(float) k/10;
      repere R(Inf(P0[1]),Sup(P0[1]),
                        Inf(P0[2]),Sup(P0[2]))
      Estim(R,P0,0.05,alpha);     // eps is set to 0.05
}
```

7 Conclusion

The problem of parameter estimation of a (non)linear model from prior knowledge, experimental data and collateral constraints is viewed in this article as one of *set inversion*, which is solved in an approximate but guaranteed way with the tools of interval analysis. It is possible to charaterize the set of all parameter vectors that are *consistent* with the data in the sense that the errors between the data and corresponding model outputs fall within known prior bounds. This has been illustrated on simple simulated examples for time-invariant models whose outputs are linear in their inputs, even if nonlinear in their parameters. it is worth stressing that the scheme of section 3 is robust in the sense

that the fit of the tuned model is at least as good as what is obtained with the classical optimization approach.
Upon completion of the algorithm, a paving bracketing the contours of the solution membership functions is found (or not) with a precision controlled by the solver.
This computation process has drawbacks: (i) its complexity is exponential in the number of parameters which restricts its use to low-dimensional problems, (ii) the algorithm presented here is far from optimal from the viewpoint of computational time and significant improvements can be expected in the near future, (iii) efficient functions are needed which are available only when an explicit solution for the equations defining the model can be found.
Realistic advantages can be found compared to the statistical approach:

1. The error structure is quite simple and similar information is usually available in most practical cases, not assuming *any a priori* statistical information about the error.
2. The computation of the parameter domain is conceptually simple and is practically feasible even if the number of data is not large.
3. The algorithm is *deterministic*.
4. *guaranteed* results are available for (non)linear models even when the parameters are not identifiable. Nonlinear constraints are easily handled.
5. the solver characteristics are different from optimization approaches that requires a (large) set of data points.

Least-Square estimation suffers from the fact that the cost function to be minimized is a sum of terms involving the same parameters, so multioccurence of these parameters is unavoidable and tends to make inclusion functions for the cost function very pessimistic, which complicates the elimination of interesting parts of the search domain.

References

1. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied interval analysis*. Springer, Paris, 2001.
2. V. Vigneron, M. Olteanu, and H. Maaref. Identification of fuzzy functions via interval analysis. *Fuzzy Sets and Systems*, 2007. to be published.
3. R.E. Moore. *Interval analysis*. Prentice Hall, New York, 1966.
4. L. Jaulin and E. Walter. Guaranteed nonlinear parameter estimation from bound-error data via interval analysis. *Mathematics and Computers in Simulation*, 35(2):123–137, 1993.
5. L. Jaulin. *Solution globale et garantie de problèmes ensemblistes. Application à l'estimation non-linéaire et à la commande robuste.* Phd thesis, Université Paris-Sud, Orsay, France, 1994.
6. J.P. Norton. Special issue on bounded-error estimation: Issue 1. In *International Symposium Control and Signal Processing*, pages 1–118, 1990.
7. E. Walter, editor. *Special issue on parameter identification with error bounds*, volume 32, 1990.
8. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Nature. *Learning representations by back-propagating errors*, 323:533–536, 1986.
9. V. Vigneron and J. Jutten. Bounded approximation for score fonction selection. In *Proceedings ICA'2003*, pages 119–124, April 2003.
10. G. Darmois. Analyse générale des liaisons stochastiques. *Rev. Inst. Intern. Stat.*, 21:2–8, 1953. (in french).
11. J. Herault and C. C. Jutten. Space ot time adaptative signal processing by neural network models. In *Neural networks for computing*, Snowbird, 1986.
12. O. Macchi and E. Moreau. Self-adaptive source separation, part i: Covergence analysis of a direct linear network controled by the herault-jutten algorithm. *IEEE Trans. on Signal Processing*, 45(4), April 1997.
13. E. Sorouchyari. Blind separation of sources, part 3: stability analysis. *Signal Processing*, 24:21–30, 1991.
14. N. Minorsky. *Nonlinear oscillations*. Reinhold, New York, 1962.