Hull Consistency Under Monotonicity

Gilles Chabert[1] and Luc Jaulin[2]

[1] Ecole des Mines de Nantes LINA CNRS UMR 6241,
4, rue Alfred Kastler 44300 Nantes, France
gilles.chabert@emn.fr
[2] ENSIETA, 2, rue Fran cois Verny 29806 Brest Cedex 9, France
luc.jaulin@ensieta.fr

Abstract. We prove that hull consistency for a system of equations or inequalities can be achieved in polynomial time providing that the underlying functions are monotone with respect to each variable. This result holds including when variables have multiple occurrences in the expressions of the functions, which is usually a pitfall for interval-based contractors. For a given constraint, an optimal contractor can thus be enforced quickly under monotonicity and the practical significance of this theoretical result is illustrated on a simple example.

1 Introduction

Solving constraint problems with real variables has been the subject of significant developments since the early 90's (see [3] for a comprehensive survey).

One of the key contribution is the concept of *hull consistency*, which is the counterpart of *bound consistency* in discrete constraint programming, as Definition 1 shows below.

Let us briefly trace the history. The underlying concepts of interval propagation appeared first in several pioneering papers [6, 11, 17, 12] while consistency techniques for numerical CSP have been formalized a few years later in [15, 5]. A theoretical comparative study of consistencies has then been conducted in [7, 8]. Finally, hull consistency has been made operational in [2, 9] where the famous HC4 algorithm is described.

Since hull consistency is based on the global consistency of every isolated constraint, enforcing hull consistency in the general case (i.e., for arbitrary nonlinear equations) is a NP-hard problem [14]. On the practical side, this results into the inability to give a sharp enclosure when variables occur more than once in the expression of a constraint. This happens, in particular, with HC4.

We show that hull consistency can be enforced in polynomial time if the functions involved are all monotone.

Monotonicity follows the very intuitive idea that a function varies either in the same direction as a variable or in the opposite one (see Definition 2). It turns out that usual functions, i.e., built with arithmetic operators (+,−,×,/) and elementary functions (sin, exp, etc.) are analytic and therefore *most of the time* strictly monotone. In rigorous terms, *most of the time* means that, unless the function is flat (or defined piecewise), the set of points that do not satisfy local strict monotonicity is of measure zero (in the sense of measure theory).

As a consequence, if a better contraction (or filtering) can be achieved under monotonicity, branch & prune algorithms should take advantage of it.

Monotonicity has been considered from the beginning of interval analysis [16], but with a motivation slightly different from ours. One of the most fundamental issues of interval analysis is the design of *inclusion functions* [13], i.e., methods for computing an enclosure of the range of a function on any given box. Of course, the sharper the better. Now, an optimal inclusion function (i.e., a method for computing the exact range on any box) can be built straightforwardly for monotone functions (see §2).

Hence, the main matter since that time has been to devise efficient way to *detect* monotonicity of a function f over a box $[x]$. One simple way to proceed is by checking that the gradient does not get null in $[x]$ which, in turn, requires an inclusion function for the gradient. The latter can then be based either on a direct interval evaluation, Taylor forms or the monotonicity test itself in a reentrant fashion (leading to the calculation of second derivatives, and so on).

Surprisingly enough, monotonicity has never been used so far in the design of *contractors*. Remember that, although related, computing a sharp enclosure for $\{f(x),\ x \in [x]\}$ and for $\{x \in [x] \mid f(x) = 0\}$ are quite different goals. As we already said, getting an optimal enclosure with a monotone function f is straightforward in the first case. But it is not so in the second case, especially when x is a vector of variables. Algorithm 1 below will provide an answer.

In the following, we first define properly the different concepts. The main result, a monotonicity-based polytime optimal contractor for a constraint, is then presented. Finally, we highlight the practical benefits of this contractor with a simple example.

1.1 Notations and definitions

We consider all along this paper a constraint satisfaction problem (CSP) with a vector of n real variables $x_1, \ldots, x_n$.

Domains of variables are represented by real intervals and a Cartesian product of intervals is called a *box*. Intervals and boxes will be surrounded by brackets, e.g., $[x]$. If $[x]$ is a box, x^- and x^+ will stand for the two opposite corners formed by the lower and upper bound respectively of each components (see Figure 2). Hence, x_i^- and x_i^+ will stand for the lower and upper bound respectively of the interval $[x]_i$. The width of an interval $[x]$ (width$[x]$) is $x^+ - x^-$.

Furthermore, given a mapping f on $\mathbb{R}^n$, we shall denote by $\{f = 0\}$ the constraint $f(x) = 0$ viewed as the set of all solution tuples, i.e.,

$$\{f = 0\} := \{x \in \mathbb{R}^n \mid f(x) = 0\}.$$

We can now give a definition to hull consistency.

Definition 1 (Hull consistency). *Let $\mathcal{P}$ be a constraint satisfaction problem involving a vector x of n variables and let $[x]$ be the domain of x.*

$\mathcal{P}$ is said to be hull consistent if for every constraint c and for all i $(1 \le i \le n)$, there exists two points in $[x]$ which satisfy c and whose i^{th} coordinates are x_i^- and x_i^+ respectively.

The key property of hull consistency lies in the combination of local reasoning and interval representation of domains. This concept brought a decisive improvement to the traditional Newton-based numerical solvers that were basically only able to contract domains *globally.*

Definition 2 (Monotonicity). *A mapping $f : \mathbb{R} \to \mathbb{R}$ is* **increasing** *over an interval $[x]$ if $\forall x \in [x], \forall y \in [x] \quad x \le y \Longrightarrow f(x) \le f(y)$.*

A mapping $f : \mathbb{R} \to \mathbb{R}$ is **decreasing** *if $-f$ is increasing, and* **monotone** *if it is either decreasing or increasing.*

A mapping $f : \mathbb{R}^n \to \mathbb{R}$ is **increasing** *(resp.* **decreasing, monotone***) over a box $[x]$ if $\forall \tilde{x} \in [x]$ and $\forall i,\ 1 \le i \le n,\ x_i \mapsto f(\tilde{x}_1, \ldots, \tilde{x}_{i-1}, x_i, \tilde{x}_{i+1}, \ldots, \tilde{x}_n)$ is monotone (resp. decreasing, monotone) over $[x]_i$.*

Strict monotonicity *is satisfied when formulas hold with strict inequalities.*

Let $f : [y] \subseteq \mathbb{R} \to \mathbb{R}$ be an increasing function. For any interval $[x] \subseteq [y]$, the infimum and the supremum of f on $[x]$ are $f(x^-)$ and $f(x^+)$. Hence, the following interval function:

$$[x] \to [f(x^-), f(x^+)]$$

is an optimal inclusion function for f. This result easily generalizes to monotone multivariate functions, by a componentwise repetition of the same argument.

2 Main result

Enforcing hull consistency on a CSP boils down to enforcing global consistency on every isolated constraint (cf. Definition 1). Giving an optimal contractor for a single constraint is thus the main issue, which we shall address below. We shall even focus on an equation $f(x) = 0$ (inequalities will be discussed further).

Consider first the univariate case (f has a single variable) and assume that f is differentiable. The set $\{f = 0\}$ can easily be bracketed by an interval Newton iteration:

$$[y] \leftarrow \Big(\tilde{y} - f(\tilde{y})/[f']([y])\Big) \cap [y]$$

where $[f']$ is a (convergent) inclusion function for f' and $\tilde{y}$ any point in $[y]$ such that $f(\tilde{y}) \neq 0$. Henceforth, we assume that a procedure `univ_newton`$(f, [x], \varepsilon)$ is available. This procedure returns an interval $[y]$ such that both $[y^-, y^- + \varepsilon]$ and $[y^+ - \varepsilon, y^+]$ intersect $\{f = 0\}$. It can refer to any implementation of the univariate interval Newton iteration, such as the one given in [10].

Let us state the complexity. As noticed in introduction, an analytic function is locally either strictly monotone or flat. Thus, it makes sense to assume strict monotonicity when dealing with complexity. The interval Newton iteration has a quadratic rate of convergence [1], i.e., the width of $[y]$ at every step is up to a constant factor less than the square of the width at the previous step. However, the quadratic rate is only achieved when the iteration is contracting, i.e., when $\tilde{y} - f(\tilde{y})/[f']([y]) \subseteq [y]$. While this condition is not fulfilled, the progression can be slow, as the following figure illustrates:

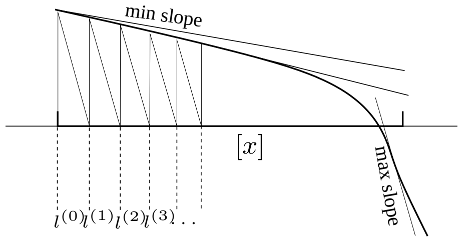


Fig. 1: Slow progression of the Newton iteration (with the left bound as point of expansion). The maximum slope on the interval $[x]$ (on the right side) is repeatedly encompassed in the interval computation of the derivative, which explains the slow progression. The successive lower bounds of the interval $[x]$ are $l^{(0)}, l^{(1)}, \ldots$.

When the point of expansion $\tilde{y}$ is the midpoint of $[y]$, the width of the interval is at least divided by two (this is somehow a way to interleave a dichotomy within the Newton iteration). Thus, the worst-case complexity of `univ_newton` with the midpoint heuristic is in $O(log(w/\varepsilon))$, where w is the width of the initial domain. Finally, note that if f is not differentiable (or if no convergent inclusion function is available for f'), one can still resort to a simple dichotomy and achieve the same complexity.

The general algorithm (called `OCTUM`: *optimal contractor under monotonicity*) that works with a multivariate mapping $f : \mathbb{R}^n \to \mathbb{R}$ is given below. Note that `univ_newton` is called on the restriction of f to (axis-aligned) edges of the input box $[x]$. Since $(n-1)$ coordinates are fixed on an edge, the restriction is indeed a function from $\mathbb{R}$ to $\mathbb{R}$.

To ease the description of the algorithm, we will assume that the multivariate function f is increasing (according to Definition 2). Once the algorithm is understood, considering the other possible configurations makes no difficulty and just require a case-by-case adaptation. Lines 1 to 5 initializes the two vectors $x^{\ominus}$ and $x^{\oplus}$ that correspond to the vertices where f is minimized and maximized respectively. When f is increasing, $x^{\ominus}$ and $x^{\oplus}$ are just aliases for x^- and x^+. Line 6 checks that the box $[x]$ contains a solution (and otherwise, the algorithm returns the empty set). The main loop relies on the following fact (see Figure 2)

that will be proved further. Remember that f is assumed to be increasing and that $[x]$ contains at least one solution. The minimum of x_i when x describes the solutions inside $[x]$ is then either reached
(1) on the edge where all the other variables are instantiated to their upper bound x_j^+ or
(2) on the face where $x_i = x_i^-$ (which means that no contraction can be made).

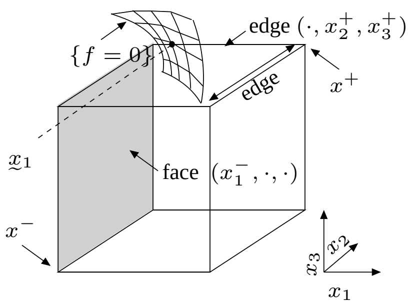


Fig. 2: The first component of the solutions inside the box either reaches its minimum on the face $(x_1^-, \cdot, \cdot)$ or on the edge $(\cdot, x_2^+, x_3^+)$.

Furthermore, as soon as the minimum for a component x_i is met on an edge, the solution on this edge makes all x_j^+ $(j \neq i)$ consistent in the domain of the j^{th} variable. To skip useless filtering operations for the upper bounds of the remaining variables, we use a flag named `sup` in the algorithm (see lines 17 and 18). Finally, when `univ_newton` is called, either the lower or upper bound of the resulting interval is considered, depending on which bound of $[x]_i$ is contracted (see lines 15 and 16 or 21 and 22). This ensures that no solution is lost.

Filtering the upper bound of x_i is entirely symmetrical. The complexity of `OCTUM` is in $O(n \times log(\frac{\text{width}[x]}{\varepsilon}))$, where width$[x]$ stands for $\max_{1 \leq i \leq n} \text{width}[x]_i$. It can be qualified as a pseudo-linear complexity.

The `OCTUM` algorithm can be very easily extended to an inequality $f(x) \leq 0$ or $f(x) \geq 0$ by simply skipping narrowing operations on y_i^- or y_i^+ respectively.

The completeness and optimality of `OCTUM` relies on the following proposition.

Proposition 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous increasing mapping and $[x]$ a box such that $\{f = 0\} \cap [x] \neq \emptyset$. Given i, $1 \leq i \leq n$ put:*

$$\underset{\sim}{x}_i := \inf\{x_i,\ x \in \{f = 0\} \cap [x]\}.$$

Then, one of the two options holds:

1. $\underset{\sim}{x}_i = x_i^-$,

Algorithm 1: OCTUM($f, [x], \varepsilon$)

```
Input: a monotone function $f$, a $n$-dimensional box $[x]$, $\varepsilon > 0$
Output: the smallest box $[y]$ enclosing $[x] \cap \{f = 0\}$, up to the precision $\varepsilon$
for $i = 1$ to $n$ do
    if $(f \nearrow x_i)$ then $x_i^{\oplus} \leftarrow x_i^+$ ;        // $x^{\oplus}$ is the vertex where $f$ is maximized
    else $x_i^{\oplus} \leftarrow x_i^-$                               // $(f \nearrow x_i)$ means "$f$ is increasing w.r.t. $x_i$"
    if $(f \nearrow x_i)$ then $x_i^{\ominus} \leftarrow x_i^-$ ;      // $x^{\ominus}$ is the vertex where $f$ is minimized
    else $x_i^{\ominus} \leftarrow x_i^+$
if $f(x^{\ominus}) > 0$ or $f(x^{\oplus}) < 0$ then return $\emptyset$     // check if a solution exists
$[y] \leftarrow [x]$
sup $\leftarrow$ false                 // true when $x_i^{\oplus}$ is consistent for all remaining $i$
inf $\leftarrow$ false                 // true when $x_i^{\ominus}$ is consistent for all remaining $i$
for $i = 1$ to $n$ do
    curr_sup $\leftarrow$ sup ;              // save the current value for the second block
    if inf is false then
        [s] $\leftarrow$ univ_newton($t \mapsto f(x_1^{\oplus}, \ldots, x_{i-1}^{\oplus}, t, x_{i+1}^{\oplus}, x_n^{\oplus}), [y]_i, \varepsilon$)
        if $[s] \neq \emptyset$ then
            if $(f \nearrow x_i)$ then $y_i^- \leftarrow s^-$          // update lower bound of $x_i$
            else $y_i^+ \leftarrow s^+$ ;                          // update upper bound of $x_i$
            sup $\leftarrow$ true                 // the edge $x_j^{\oplus}$, $j \neq i$, contains a solution
    if curr_sup is false then
        [s] $\leftarrow$ univ_newton($t \mapsto f(x_1^{\ominus}, \ldots, x_{i-1}^{\ominus}, t, x_{i+1}^{\ominus}, x_n^{\ominus}), [y]_i, \varepsilon$)
        if $[s] \neq \emptyset$ then
            if $(f \nearrow x_i)$ then $y_i^+ \leftarrow s^+$
            else $y_i^- \leftarrow s^-$
            inf $\leftarrow$ true
return $[y]$
```

2. there exists $x \in [x]$ such that $x_i = \underset{\sim}{x}_i$ and for every $j \neq i$, $x_j = x_j^+$.

Proof. Let x^* be a solution point in $[x]$ minimizing x_i, i.e., $f(x^*) = 0$ and $x_i^* = \underset{\sim}{x}_i$. If $x_i^* = x_i^-$, the first option holds and we are done. Assume $x_i^* > x_i^-$. If $n = 1$, the second option holds trivially. If $n > 1$, consider the following vector:

$$\overline{x} := (x_1^+, \ldots, x_{i-1}^+, x_i^*, x_{i+1}^+, \ldots, x_n^+).$$

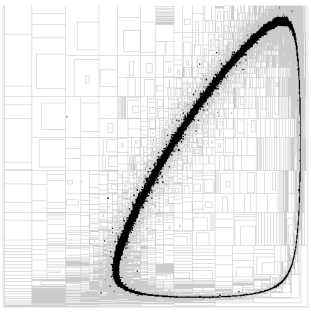
We will prove that $f(\overline{x}) = 0$. By contradiction, assume $f(\overline{x}) > 0$ (f being increasing). Since $\overline{x}_i = x_i^* > x_i^+$, there exists by continuity ε, $0 < \varepsilon < 1$, such that $f(\overline{x} - \varepsilon e_i) > 0$. Now, x^* is the point in $[x]$ where f gets null with the smallest i^{th} coordinate. Hence, $f(x^* - \varepsilon e_i) < 0$.

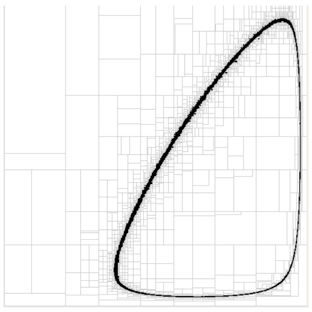
Since f is continuous, f gets null somewhere on the segment joining $x^* - \varepsilon e_i$ and $\overline{x} - \varepsilon e_i$ because the sign of f is opposite at the two extremities. Since $[x]$ is convex, the corresponding point is inside $[x]$ and its i^{th} component is $x_i^* - \varepsilon_i$, which contradicts the fact that x_i^* is the infimum among the solutions. ▲

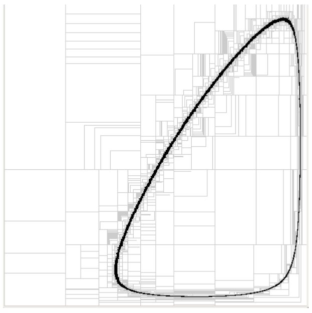
3 A first experiment

Let us consider the problem of characterizing the set of points (x, y) in $[-3, 0] \times [0, 3]$ satisfying $f(x, y) = 0$ with $f(x, y) = x^2y^2 - 9x^2y + 6xy^2 - 20xy - 1$.

Let us compare `OCTUM` with three other standard generic contractors, namely `HC4` [2, 9], `BOX` [4, 18] and `3B` [15]. We have implemented a very naive method for detecting monotonicity, using a simple interval evaluation of the gradient that is systematically computed for every box (a better method would be to manage flags w.r.t. each variable in an *backtrackable* structure, each flag being set incrementally as soon as f is proven to be increasing or decreasing). Even with this naive implementation, `OCTUM` yields better results, either in terms of *quality* (see Figure 3) or *quantity* (see Table below).


(a) Using `HC4`.


(b) Using `BOX`.


(c) Using `3B`.

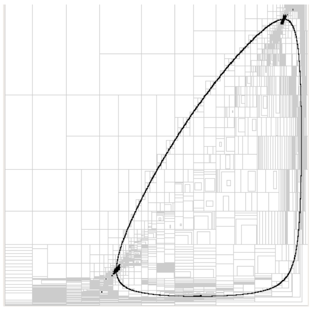

(d) Using `OCTUM`.

Fig. 3: Comparing the monotonicity-based contractor `OCTUM` with other standard operators. The resulting paving is more tight to the actual solution set. Note also that he two little stains in (d) correspond to the area where the gradient of f is close to zero.

	Running time	Number of backtracks	Size of solution set
`HC4`	0.66s	28240	6928
`BOX`	1.37s	9632	3595
`3B`	1.89s	9171	2564
`OCTUM`	**0.40s**	**6047**	**1143**

4 Conclusion

We have proven that hull consistency can be achieved in polynomial time in the case of constraints involving monotone functions. Hull consistency amounts to global consistency for each isolated constraint. We have given an algorithm called `OCTUM` that enforces global consistency for an equation under monotonicity (and explained how to adapt it to inequalities). Hull consistency based on `OCTUM` can then be programed by simply embedding `OCTUM` in a classical AC3 propagation loop. A first experiment has illustrated the two nice properties of `OCTUM`: optimality and (pseudo-)linear complexity.

References

1. G. Alefeld and G. Mayer. Interval Analysis: Theory and Applications. *J. Comput. Appl. Math.*, 121(1-2):421–464, 2000.
2. F. Benhamou, F. Goualard, L. Granvilliers, and J-F. Puget. Revising Hull and Box Consistency. In *ICLP*, pages 230–244, 1999.
3. F. Benhamou and L. Granvilliers. Continuous and interval constraints. In *Handbook of Constraint Programming*, chapter 16, pages 571–604. Elsevier, 2006.
4. F. Benhamou, D. McAllester, and P. Van Hentenryck. CLP(intervals) revisited. In *International Symposium on Logic programming*, pages 124–138. MIT Press, 1994.
5. F. Benhamou and W.J. Older. Applying Interval Arithmetic to Real, Integer and Boolean Constraints. *Journal of Logic Programming*, 32:1–24, 1997.
6. J.G. Cleary. Logical Arithmetic. *Future Computing Systems*, 2(2):125–149, 1987.
7. H. Collavizza. A Note on Partial Consistencies over Continuous Domains Solving Techniques. In *CP*, pages 147–161, 1998.
8. F. Delobel, H. Collavizza, and M. Rueher. Comparing Partial Consistencies. *Reliable Computing*, 5(3):213–228, 1999.
9. L. Granvilliers and F. Benhamou. Progress in the Solving of a Circuit Design Problem. *Journal of Global Optimization*, 20(2):155–168, 2001.
10. E.R. Hansen. *Global Optimization using Interval Analysis.* Marcel Dekker, 1992.
11. E. Hyvönen. Constraint Reasoning Based on Interval Arithmetic. In *IJCAI*, pages 1193–1198, 1989.
12. E. Hyvönen. Constraint Reasoning Based on Interval Arithmetic—The Tolerance Propagation Approach. *Artificial Intelligence*, 58:71–112, 1992.
13. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis.* Springer, 2001.
14. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl. *Computational complexity and feasibility of data processing and interval computations.* Kluwer, 1997.
15. O. Lhomme. Consistency Techniques for Numeric CSPs. In *IJCAI*, pages 232–238, 1993.
16. R. Moore. *Interval Analysis.* Prentice-Hall, 1966.
17. W.J. Older and A. Vellino. Extending Prolog with Constraint Arithmetic on Real Intervals. In *IEEE Canadian Conf. on Elec. and Comp. Engineering*, 1990.
18. P. Van Hentenryck, D. McAllester, and D. Kapur. Solving Polynomial Systems Using a Branch and Prune Approach. *SIAM J. Numer. Anal.*, 34(2):797–827, 1997.