A New Framework for Sharp and Efficient Resolution of NCSP with Manifolds of Solutions

Alexandre Goldsztejn[1] and Laurent Granvilliers[2]

[1] CNRS, LINA, UMR 6241
Alexandre.Goldsztejn@univ-nantes.fr
[2] Université de Nantes, Nantes Atlantique Université, CNRS, LINA, UMR 6241
Laurent.Granvilliers@univ-nantes.fr

Abstract. When numerical CSPs are used to solve systems of n equations with n variables, the interval Newton operator plays a key role: It acts like a global constraint, hence achieving a powerful contraction, and proves rigorously the existence of solutions. However, both advantages cannot be used for under-constrained systems of equations, which have manifolds of solutions. A new framework is proposed in this paper to extend the advantages of the interval Newton to under-constrained systems of equations. This is done simply by permitting domains of CSPs to be parallelepipeds instead of the usual boxes.

1 Introduction

The paper presents a new framework for solving numerical CSPs formed of under-constrained systems of equations:

$$\langle\ \mathbf{x}\ ,\ \mathbf{f}(\mathbf{x}) = \mathbf{0}\ ,\ [\mathbf{x}]\ \rangle, \tag{1}$$

where vectorial notations[1] are used, i.e. $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of variables, $\mathbf{f} = (f_1, \ldots, f_m)$, with $m < n$, is a vector of functions and $[\mathbf{x}] = ([x_1], \ldots, [x_n])$ is a vector of interval domains. These NCSPs arise naturally in a wide range of applications, among which are surface intersection characterization and plotting [1], a particular case of implicit equation solving [2], global optimization [3] and robots kinematics [4].

Algorithms designed for well constrained NCSP [5,6] are not efficient for solving (1). Indeed, these algorithms are variations of the branch and prune algorithm where the (preconditioned) interval Newton operator [7] plays a key role: On the one hand, it acts like a global constraint that performs powerful contraction when the domains become small enough. On the other hand, it can prove rigorously the existence of a solution of a well constrained system of equations. However, these two key contributions of the interval Newton operator are not operating when dealing with under-constrained systems of equations, mainly because no preconditioning of these systems of equations has been proposed yet. The main

[1] Vectors of reals and vectors of functions are represented with boldface symbols.

© Springer-Verlag Berlin Heidelberg 2008

contribution of this paper is to present a framework that extends these two advantages of the interval Newton operator to NCSPs formed of under-constrained systems of equations.

In the usual definition of a CSP, each variable is given a domain. As a matter of fact, it is equivalent to consider the Cartesian product of these variable domains (which is a box when variable domains are intervals) as a search space for a vector made of the CSP variables. Then, more complicated sets, which are not anymore the Cartesian product of variable domains, can be used. The framework proposed in this paper shows that using parallelepiped domains instead of box domains can drastically improve the efficiency of branch and prune algorithms dedicated to (1). On the one hand, parallelepipeds offer a more flexible description of subsets of $\mathbb{R}^n$ than boxes, hence providing a more accurate enclosure of the NCSP solution set. On the other hand, using parallelepiped domains introduces an efficient preconditioning process for under-constrained systems of equations, hence allowing the interval Newton operator to both work as a global constraint and prove the existence of solutions.

Parallelepipeds have already been used to advantageously replace boxes, for example to rigorously enclose the solutions of initial values problems (cf. the survey paper [8] and references therein). The relationship between parallelepipeds and preconditioning has already been investigated in [9,10], where parallelepipeds are used to approximate the solution set of linear systems with interval uncertainties. However, the ways and means of the introduction of parallelepiped domains in the present work are totally different than in [9,10]. Finally, parallelepipeds have been used in conjunction with existence theorems to build inner approximations of function ranges in [11], but again in the restricted case of well-constrained systems of equations.

Outline. The framework proposed in this paper is presented on a motivating example in Section 2. Then, the concepts of interval analysis used in this paper are given in Section 3. The technical description of the proposed branch and prune algorithm is given in Section 4. Finally, promising experiments are presented in Section 5.

2 A Motivating Example

The usefulness of the usage of parallelepiped domains instead of box domains is now illustrated on a simple example.

2.1 Contractions and Bisections Using Box Domains

Let us consider the very simple under-constrained CSP

$$\langle\ \mathbf{x}\ ,\ \{f(\mathbf{x})=0\}\ ,\ [\mathbf{x}]\ \rangle, \tag{2}$$

with $\mathbf{x}=(x_1,x_2)$, $f(\mathbf{x})=x_1^2+x_2^2-1$ and $[\mathbf{x}]=([0.3,0.7],[0.6,1.0])$. Its solution set is plotted in Figure 1-(a).

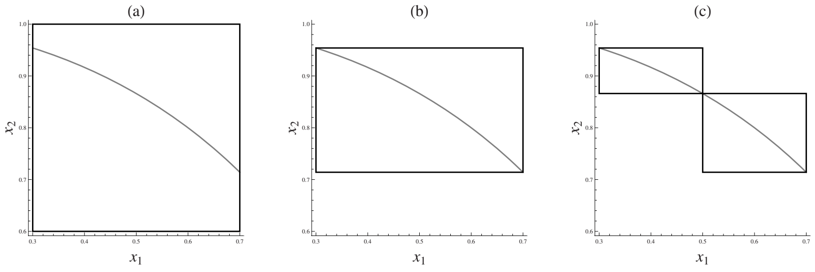

Fig. 1. (a) Solution set of the CSP (2). (b) Domain after one contraction. (c) Domains after one bisection and two more contractions.

The key point is to notice that the contraction and bisection processes of a branch and prune algorithm are not efficient in this situation: A contraction[2] is performed in Figure 1-(b), and two more contractions are performed after a bisection in Figure 1-(c). These plots clearly show that these contractions are not efficient because the solution set crosses the box domain in its diagonal. The contraction/bisection would be much more efficient if the solution set crossed the box domain along one of the axes. Reaching this situation is the goal of using parallelepiped domains instead of box domains.

2.2 From a Box Domain to a Parallelepiped Domain

To this end, a parallelepiped[3] is built whose two axes are respectively approximately parallel and perpendicular to the solution set. These directions are related to the gradient of the function f evaluated at the midpoint of the box domain. Once the parallelepiped axes are computed, the parallelepiped is chosen as small as possible under the constraint that it contains the original box domain (cf. Figure 2-(a) where the former box domain is represented using dashed lines).

Note that changing the box domain to an enclosing parallelepiped domain introduces new solutions on the border of the parallelepiped domain (the solutions that are inside the parallelepiped domain but outside the former box domain). In order to reject these additional solutions (which otherwise would be redundant with the neighbor domains), the former box domain is reintroduced as four inequality constraints which are added to the constraint store.

To see how a parallelepiped domain can improve the contraction/bisection process, we have to formalize its definition: This parallelepiped is the image of a box through an affine map $\mathbf{u} \mapsto C \cdot \mathbf{u} + \tilde{\mathbf{x}}$, i.e.

$$\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}, \tag{3}$$

[2] In this introducing example, the best contractions are performed. In practice, contractions are not that efficient. Nevertheless, this illustrates the best that can be obtained from both methods.

[3] Note that in this motivating example parallelepipeds have perpendicular axes, but this is not the case in general for higher dimensions.

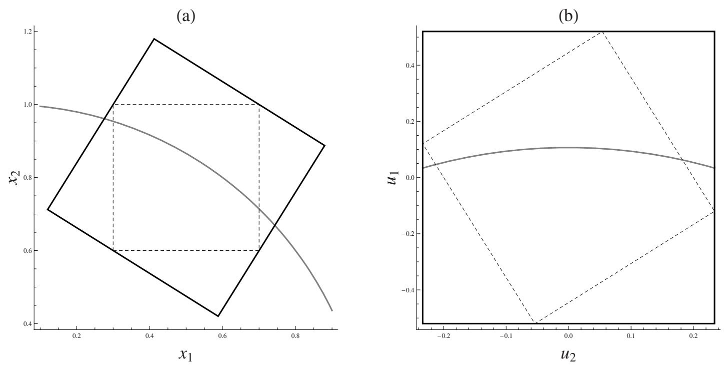

Fig. 2. (a) Enclosing parallelepiped domain for the CSP (2). (b) The same CSP expressed in the auxiliary basis formed of the parallelepiped axes.

where

- The matrix $C \in \mathbb{R}^{2\times 2}$ is a square matrix. This matrix gives its shape to the parallelepiped and is chosen so that the solution set is approximately parallel to one of its sides. This is done considering the gradient of the function evaluated at the midpoint of the box (cf. Section 4 for more details).
- The vector $\tilde{\mathbf{x}}$ is the midpoint of the box $[\mathbf{x}]$.
- The box $[\mathbf{u}]$ is computed in such a way that the parallelepiped encloses the box domain $[\mathbf{x}]$, i.e. $[\mathbf{u}] = C^{-1} \cdot ([\mathbf{x}] - \tilde{\mathbf{x}})$ where interval arithmetic is used (cf. Section 3).

Then, the CSP (2) can be expressed in the auxiliary basis formed of the characteristic axes of the parallelepiped, giving rise to the auxiliary CSP

$$\langle\ \mathbf{u}\ ,\ \{g(\mathbf{u}) = 0\}\ ,\ [\mathbf{u}]\ \rangle, \tag{4}$$

with $g(\mathbf{u}) = f(C \cdot \mathbf{u} + \tilde{\mathbf{x}})$, that is explicitly

$$g(u_1, u_2) = f(C_{11}u_1 + C_{12}u_2 + \tilde{x}_1, C_{21}u_1 + C_{22}u_2 + \tilde{x}_2). \tag{5}$$

The solution sets of (4) is represented by Figure 2-(b). As mentioned previously, four linear inequalities are added to the constraints of (4) which represent the belonging to the original box domain (they are not given explicitly in (4) for clarity). These inequalities are represented using dashed lines in Figure 2-(b). Note that the solution sets of (2) and (4) are closely related: The former is exactly the image of the latter through the affine transformation $\mathbf{u} \mapsto C \cdot \mathbf{u} + \tilde{\mathbf{x}}$.

The next two subsections show how to contract and bisect this parallelepiped domain.

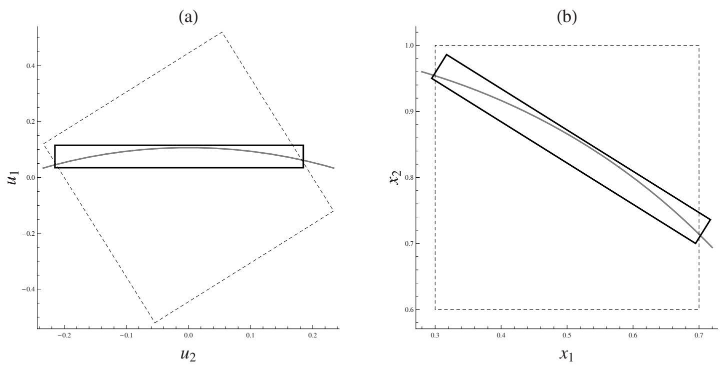

Fig. 3. Contracting parallelepiped domains

2.3 Contracting Parallelepiped Domains

Contracting the parallelepiped $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ consists in contracting its characteristic domain $[\mathbf{u}]$. The aim is to keep all the solutions of the original CSP (2) within the contracted parallelepiped. This is obviously equivalent to contracting $[\mathbf{u}]$ without losing any solution of the auxiliary CSP (4). As this later CSP has a box domain, one can use the usual techniques dedicated to NCSPs. Note that since the solution set crosses the parallelepiped in the direction u_2, the constraint $g(\mathbf{u}) = 0$ will contract efficiently the domain $[u_1]$ but will certainly be useless for the domain $[u_2]$. On the other hand, the inequality constraints coming from the box domain will help contracting the domain $[u_2]$. The contraction of $[\mathbf{u}]$ obtained for this introducing example is shown in Figure 3. Figure 3-(a) shows how $[\mathbf{u}]$ is contracted using the auxiliary CSP (4) while Figure 3-(b) shows the corresponding contraction for the parallelepiped domain of the original CSP (2). Comparing Figure 3-(b) to Figure 1-(b) shows how much more efficient the contraction of the parallelepiped domain is compared to the contraction of the original box domain.

The auxiliary CSP (4) is actually more complicated than the original CSP (2): The function $g(\mathbf{u}) = f(C \cdot \mathbf{u} + \tilde{\mathbf{x}})$ contains more occurrences of each variables (cf. Equation (5)), which is well known to decrease the efficiency of interval based methods. However, this situation is very similar to the preconditioning of the interval Newton operator. And indeed, this acts like a *right-preconditioning*[4] where the interval Newton operator should be very efficient. Actually, Section 4 shows that this right-preconditioning process allows the interval Newton to both act like a global constraint, and rigorously prove the existence of the manifold of solutions. In the context of this introducing example, the interval Newton operator proves that for all $u_2 \in [u_2]$ there exists $u_1 \in [u_1]$ such that $g(\mathbf{u}) = 0$, hence proving that the auxiliary solution set (4) crosses $[\mathbf{u}]$ along u_2. Therefore, the original (2) is proved to cross the parallelepiped in this direction.

[4] I.e. a preconditioning where the change of basis is applied before the function, see e.g. [9,10,11].

2.4 Bisecting Parallelepiped Domains

Bisecting a parallelepiped $\{C\cdot\mathbf{u}+\tilde{\mathbf{x}} : \mathbf{u}\in[\mathbf{u}]\}$ into two smaller parallelepipeds is naturally done by bisecting its characteristic domain $[\mathbf{u}]$ (cf. Figure 4-(a)), thus obtaining two new parallelepipeds $\{C\cdot\mathbf{u}+\tilde{\mathbf{x}} : \mathbf{u}\in[\mathbf{u}']\}$ and $\{C\cdot\mathbf{u}+\tilde{\mathbf{x}} : \mathbf{u}\in[\mathbf{u}'']\}$ (cf. Figure 4-(b)). Note that it obviously makes no sense to bisect $[\mathbf{u}]$ to $([u_1'],[u_2])$ and $([u_1''],[u_2])$, which would not preserve the solution set transverse crossing. Instead, bisecting $[\mathbf{u}]$ to $([u_1],[u_2'])$ and $([u_1],[u_2''])$ does preserve this transversality, and is therefore a very efficient bisection heuristic. This bisection heuristic will be trivially extended to the general case of arbitrary dimensions.

Once bisected, their characteristic matrices are updated using the same process as described previously, but based on the gradient vector of f evaluated at the center of each new parallelepiped. This allows the new parallelepipeds adapting their shape more accurately to the shape of the solution set (cf. Figure 4-(c)). Furthermore, as done with the original box domain, each former parallelepiped domain is expressed as four additional inequality constraints in its CSP (represented in dashed lines in Figure 4-(c)), which will be used to reduce the overlapping introduced when updating the characteristic matrices of the new parallelepipeds. Finally, a contraction is performed on these two new parallelepipeds, leading to Figure 4-(d). Comparing this figure to Figure 1-(c)

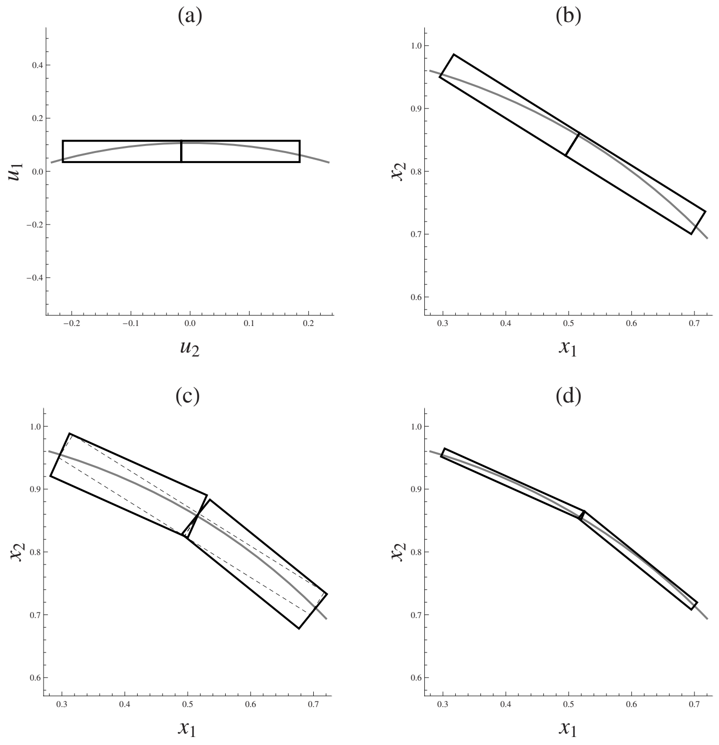

Fig. 4. Bisecting parallelepiped domains

shows how important is the improvement obtained using parallelepipeds domains instead of box domains.

The gains in contracting efficiency and existence proving illustrated on this motivating example are even more important when dealing with more complicated NCSPs of higher dimension (c.f. subsections 5.3 and 5.4).

3 Interval Analysis for NCSP Resolution

The modern interval analysis was born in the 60's with [12]. Since, it has been widely developed and is today one central tool in the resolution of constraints acting over continuous domains (see [13] and extensive references).

3.1 Interval Arithmetic

Intervals, interval vectors and interval matrices are denoted using brackets. Their set are denoted respectively by $\mathbb{IR}$, $\mathbb{IR}^n$ and $\mathbb{IR}^{n\times m}$. The elementary function are extended to intervals in the following way: let $\circ \in \{+,-,\times,/\}$ then $[x]\circ[y]=\{x\circ y : x\in[x], y\in[y]\}$ (division is defined only for non zero containing interval denominators). E.g. $[a,b]+[c,d]=[a+c,b+d]$. Also, continuous one variable functions $f(x)$ are extended to intervals using the same definition: $f([x])=\{f(x) : x\in[x]\}$, which is an interval because f is continuous. When one represents numbers using a finite precision, the previous operations cannot be computed in general. The outer rounding is then used so as to keep valid the interpretations. For example, $[1,2]+[2,3]$ would be equal to $[2.999,5.001]$ if rounded with a three decimal accuracy.

Then, an expression which contains intervals can be evaluated using this interval arithmetic. The main property of interval analysis is that such an interval evaluation gives rise to a superset of the image through the function of the interval arguments: For example, $[x]\times([y]-[x])\supseteq\{x(y-x) : x\in[x], y\in[y]\}$. In some cases (e.g. when the expression contains only one occurrence of each variable), this enclosure is optimal. In particular, the computation $C\cdot[\mathbf{u}]+\tilde{\mathbf{x}}$ is the smallest box that contains the parallelepiped $\{C.\mathbf{u}+\tilde{\mathbf{x}} : u\in[\mathbf{u}]\}$.

3.2 Interval Contractors

Given an n-ary constraint c and a box $[\mathbf{x}]\in\mathbb{R}^n$, a contractor for c will contract the box $[\mathbf{x}]$ without losing any solution of c. Some widely used contractors are based on the 2B-consistency (also called hull-consistency) or the box consistency [14,15], which are adaptations of the arc-consistency to continuous domains. They are both applied to one constraint at a time, hence suffering of the usual drawbacks of the locality of their application. On the other hand, the preconditioned interval Newton [7] can be applied to a set of n equations and n variables. Under some hypothesis on the Jacobian matrix of the system evaluated on the domain of the CSP, it will be able to treat this set of constraint as a global constraint and hence achieve a powerful contraction. Furthermore, the interval Newton can rigorously prove the existence of a solution in a CSP domain. These two characteristics of the interval Newton makes it a key tool for the resolution of NCSPs, however previously restricted to well-constrained systems of equations.

4 Description of the Algorithm

The branch and prune Algorithm 1 used here is classical except for line 1 which will be explained in the rest of the paper: The input is a CSP $\mathcal{P}$ and the output a set of CSPs $\mathcal{L} = \{\mathcal{P}_1, \ldots, \mathcal{P}_s\}$ whose disjunction is equivalent to the original CSP. Formally,

$$\mathrm{Sol}(\mathcal{P}) = \bigcup_{Q \in \mathcal{L}} \mathrm{Sol}(\mathcal{Q}). \tag{6}$$

Normally, the set of CSPs $\mathcal{L}$ is a more accurate description of the solution set that the original CSP: First, the union of their domains is much smaller than the original domain, hence providing a sharper enclosure. Second, it is often possible to prove that these CSPs actually contains some solutions, which is a crucial information. The solution existence proof is not explicitly described in Algorithm 1 for clarity. Informally, some existence proof can result of the contraction performed at Line 1 when it is computed using the interval Newton operator. When the existence of solutions is proved, this information is attached to the CSP.

When the algorithm starts, the domain of the CSP is a box. The function UpdateDomainShape at Line 1 allows changing the shape of the domain (changing a box to a parallelepiped when first successfully applied, or a parallelepiped to another parallelepiped more suited to the shape of the solution set). The first change from a box to a parallelepiped is performed only when the shape of the solution set can be foreseen from the evaluation of the constraints derivatives (cf. Subsection 4.1), which implies that the box domain is small enough. As a consequence, the algorithm will use box domains in a first phase, and parallelepiped domains as soon as box domains are small enough to identify the directions of the solution manifold. While the CSP domain is a box, usual contractors are used to prune its domain (2B-consistency based contractors and non-preconditioned interval Newton operator in our implementation of Algorithm 1, whose collaboration is known to be efficient). When the domain is changed to a parallelepiped, the interval Newton operator is adapted and allows powerful contractions and existence proof of solutions (cf. Subsection 4.2). Finally, parallelepiped domains are bisected similarly to box domains (cf. Subsection 4.3).

4.1 Changing the Shape of a Parallelepiped

The function UpdateDomainShape($\mathcal{P}$) attempts to find a new parallelepiped domain $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ that will be more suited to the solution set of $\mathcal{P}$. The former domain of $\mathcal{P}$ is the parallelepiped $\{D \cdot \mathbf{v} + \tilde{\mathbf{x}} : \mathbf{v} \in [\mathbf{v}]\}$, which is possibly a box in which case $D = \mathrm{id}$ (note that the former and the new

Algorithm 1. Branch and prune algorithm using parallelepiped domains

```
Input: $\mathcal{P} = \langle \mathbf{x}, \mathcal{C}, [\mathbf{x}] \rangle$, $\epsilon$
Output: $\mathcal{L} = \{\mathcal{P}_1, \dots, \mathcal{P}_s\}$
$\mathcal{T} \leftarrow \{\mathcal{P}\}$; $\mathcal{L} \leftarrow \emptyset$;
while ( not $\mathcal{T} = \emptyset$ ) do
    $\mathcal{P} \leftarrow$ Extract($\mathcal{T}$);
    if ( Measure(Domain($\mathcal{P}$)) $> \epsilon$ ) then
        $\mathcal{P}' \leftarrow$ UpdateDomainShape($\mathcal{P}$);
        $\mathcal{P}'' \leftarrow$ Contract($\mathcal{P}'$);
        if ( Domain($\mathcal{P}''$) $\neq \emptyset$ ) then
            $(\mathcal{Q}, \mathcal{Q}') \leftarrow$ Bisect($\mathcal{P}''$);
            $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{Q}, \mathcal{Q}'\}$;
        end
    else
        $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{P}\}$;
    end
end
return ($\mathcal{L}$);
```

parallelepiped domains share the same characteristic vector $\tilde{\mathbf{x}}$). This process is done in two steps: First a candidate new parallelepiped domain $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ is computed. Then, the efficiency of this candidate parallelepiped domain is verified a posteriori. If not useful, the former parallelepiped is kept. If the candidate parallelepiped domain is chosen, then some inequality constraints are added to the CSP in order to prevent some parasite solutions to appear due to the enclosure of the former parallelepiped domain inside a new parallelepiped (cf. Section 2).

Computation of the Candidate Parallelepiped Domain. As illustrated in Section 2, the aim of using a new parallelepiped domain $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ is to apply the interval Newton operator in the auxiliary basis of the parallelepiped to reduce directly the box domain $[\mathbf{u}]$. Hence, the parallelepiped characteristic matrix C is chosen aiming an efficient application of the interval Newton operator. The box domain $[\mathbf{u}]$ will be reduced using the constraint $\mathbf{g}(\mathbf{u}) = \mathbf{0}$ where $\mathbf{g}(\mathbf{u}) = \mathbf{f}(C \cdot \mathbf{u} + \tilde{\mathbf{x}})$ (cf. Section 2). To obtain an efficient application of the interval Newton operator, the under-constrained system $\mathbf{g}(\mathbf{u}) = \mathbf{0}$ needs to be interpreted as a system of equations m equations and m variables, where the remain $n - m$ variables are considered as parameters. Then, to be efficient the interval Newton operator requires the Jacobian $J\mathbf{g}$ of $\mathbf{g}$ to be close to the matrix $\left(I_{m\times m} \mid 0_{m\times(n-m)} \right)$, where $I_{m\times m}$ is the identity matrix of size $m \times m$ and $0_{m\times(n-m)}$ is the null matrix of size $m \times (n - m)$. As $J\mathbf{g} = J\mathbf{f} \cdot C$, it is natural to choose C such that

$$J\mathbf{f}([\mathbf{x}]) \cdot C \;\approx\; \left(I_{m\times m} \mid 0_{m\times(n-m)} \right), \tag{7}$$

where $[\mathbf{x}]$ is the interval hull of the parallelepiped domain, i.e. $[\mathbf{x}] = C \cdot [\mathbf{u}] + \tilde{\mathbf{x}}$.

To this end, C is constructed based on the evaluation of the Jacobian of $\mathbf{f}$ at $\tilde{\mathbf{x}}$. The matrix C^{-1} is first constructed as follows, and C will be obtained inverting C^{-1}. The i^{th} line $(C^{-1})_i$ of C^{-1} is defined by:

– The gradient of f_i evaluated at $\tilde{\mathbf{x}}$ if $i \leq m$.
– A vector orthogonal to all previously computed $(C^{-1})_k$ for $1 \leq k < i$ (these vectors are not uniquely determined, but a set of such vectors is easily obtain using a Gram-Schmidt orthogonalization).

Then, this matrix is inverted[5] to obtain C. The matrix C thus satisfies $J\mathbf{f}(\tilde{\mathbf{x}})\cdot C = \big(I_{m\times m} \,\big|\, 0_{m\times(n-m)} \big)$, up to rounding errors (indeed by construction, $J\mathbf{f}(\tilde{\mathbf{x}}) \cdot C$ is made of the m first rows of the $n \times n$ identity matrix).

Finally, the characteristic domain $[\mathbf{u}]$ is computed so that the new parallelepiped domain encloses the former: $[\mathbf{u}] = (C^{-1}D)[\mathbf{v}]$. This time, interval arithmetic is used to ensure a rigorous enclosure.

Verification of the Efficiency of the Candidate New Parallelepiped Domain. Formally, the interval Newton will be efficient if the square matrix formed of the first m columns of $J\mathbf{f}([\mathbf{x}]) \cdot C$ is diagonally dominant (cf. Theorem 5.2.5 in [7]). Therefore, the interval matrix $J\mathbf{f}([\mathbf{x}]) \cdot C$ is computed explicitly, and the diagonal dominance of its first m columns is checked. If it is diagonally dominant, the parallelepiped domain is updated, and some inequality constraints are added to the CSP (cf. the next section). Otherwise, the former parallelepiped domain is kept, i.e. $C = D$ and $[\mathbf{u}] = [\mathbf{v}]$.

Remark 1. If $\mathbf{x}$ is a solution (i.e. $\mathbf{f}(\mathbf{x}) = \mathbf{0}$) and is singular (i.e. $D\mathbf{f}(\mathbf{x})$ is not full rank, e.g. has two proportional lines) then $J\mathbf{f}([\mathbf{x}]) \cdot C$ will never be diagonally dominant. In this case, the algorithm keeps working with box domains around this point. This situation is untypical as some arbitrary small perturbation of the problem can change the singular solutions to regular solutions.

Adding Linear Inequalities to the new CSP. As illustrated in Section 2, the enclosure of the former parallelepiped domain by a new parallelepiped domain is not perfect. Hence, some solutions that are inside the new domain may not be in the former, which would lead to redundant solutions if not properly treated. To this end, the former parallelepiped domain is reintroduced in the new CSP as $2n$ linear constraints:

$$\underline{v}_i \;\leq \sum_{1\leq j\leq n} D_{ij}x_j \;\leq \overline{v}_i, \tag{8}$$

for $1 \leq i \leq n$. As a consequence, the former and the new CSPs have the same solution set. These linear inequalities will be denoted using the scalar product notation $\mathbf{a}\cdot\mathbf{x} \leq b$ where $\mathbf{a} = (D_{i1},\ldots,D_{in})$ and $b = \overline{v}_i$, or $\mathbf{a} = -(D_{i1},\ldots,D_{in})$ and $b = -\underline{v}_i$.

[5] Note that C needs only to be computed approximately and thus standard double precision computations can be used at this step.

4.2 Contracting a Parallelepiped Domain

When the domain of the CSP is a box, the usual techniques are used to contract it. When it is a parallelepiped, two kind of constraints are in the store: the linear inequalities $\mathbf{a} \cdot \mathbf{x} \leq b$ and the nonlinear equalities $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.

Linear Inequalities. In order to contract the characteristic domain $[\mathbf{u}]$ of the parallelepiped $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ under the linear inequality $\mathbf{a} \cdot \mathbf{x} \leq b$, this constraint is simply expressed in the basis of the parallelepiped:

$$\mathbf{a} \cdot \mathbf{x} \leq b \iff (\mathbf{a} \cdot C) \cdot \mathbf{u} \leq b - \mathbf{a} \cdot \tilde{\mathbf{x}}. \tag{9}$$

Then, $[\mathbf{u}]$ is contracted under the new linear inequality using the 2B-consistency.

Nonlinear Equalities. The nonlinear equalities are also expressed in the basis of the parallelepiped: $\mathbf{f}(\mathbf{x}) = \mathbf{0} \iff \mathbf{g}(\mathbf{u}) = \mathbf{0}$, with $\mathbf{g}(\mathbf{u}) = \mathbf{f}(C \cdot \mathbf{u} + \tilde{\mathbf{x}})$. Then the m first components of $[\mathbf{u}]$ are contracted considering this under-constrained system of equations as a well constrained parametric systems of equations. Let $\mathbf{u}' = (u_1, \ldots, u_m)$ be the vector of the first m variables, and $\mathbf{u}'' = (u_{m+1}, \ldots, u_n)$ be the vector of the remaining variables, which are considered as parameters. The interval Newton is then applied to the parametric system of equation (see [16] for details) to contract $[\mathbf{u}']$ to a smaller box $[\tilde{\mathbf{u}}']$

$$[\tilde{\mathbf{u}}'] \; = \; \hat{\mathbf{u}}' + \Gamma([J_{\mathbf{u}'}], [\mathbf{u}'] - \hat{\mathbf{u}}', \mathbf{b}) \tag{10}$$

with $\mathbf{b} = -\mathbf{f}(C \cdot \mathbf{u} + \tilde{\mathbf{x}}) - [J_{\mathbf{u}''}] \cdot ([\mathbf{u}''] - \hat{\mathbf{u}}'')$ where $[J_{\mathbf{u}'}]$ and $[J_{\mathbf{u}''}]$ are respectively the square interval matrix formed of the first m columns of $J\mathbf{f}(C.[\mathbf{u}] + \tilde{\mathbf{x}}) \cdot C$ and the rectangular matrix formed of the remaining columns. The operator Γ is the interval Gauss-Seidel method [7]. The vectors $\hat{\mathbf{u}}'$ and $\hat{\mathbf{u}}''$ are the midpoint of the respective boxes.

As the interval matrix $[J_{\mathbf{u}'}]$ is centered on the identity matrix and diagonally dominant (cf. Section 4.1), the interval Newton operator may be strictly contracting (i.e. $[\tilde{\mathbf{u}}']$ is included inside the interior of $[\mathbf{u}']$) and hence is able to prove the existence of solutions. In this case, the following existence statement holds [16]: $\forall \mathbf{u}'' \in [\mathbf{u}''], \exists \mathbf{u}' \in [\mathbf{u}'], \mathbf{g}(\mathbf{u}) = \mathbf{0}$, hence proving that the solution set crosses the whole parallelepiped domain transversally. When the existence is proved, this information is recorded together with the CSP.

4.3 Bisecting a Parallelepiped Domain

A parallelepiped domain $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}} : \mathbf{u} \in [\mathbf{u}]\}$ is bisected splitting $[\mathbf{u}]$. However, as the contraction of $[\mathbf{u}']$ performed using the interval Newton is convergent, it is useless to bisect these components of the box. Therefore, $[\mathbf{u}]$ is bisected to $[\tilde{\mathbf{u}}_1]$ and $[\tilde{\mathbf{u}}_2]$ where the largest component of $\mathbf{u}''$ has been bisected. This is sufficient to ensure the convergence of the algorithm by Theorem 5.2.5 in [7]. Finally, the two bisected parallelepipeds $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}}_1 : \mathbf{u} \in [\mathbf{u}_1]\}$ and $\{C \cdot \mathbf{u} + \tilde{\mathbf{x}}_2 : \mathbf{u} \in [\mathbf{u}_2]\}$ are defined as follows: for both $k = 1$ and $k = 2$, $\tilde{\mathbf{x}}_k = C \cdot \mathrm{mid}([\tilde{\mathbf{u}}_k]) + \tilde{\mathbf{x}}$ so that this vector is at the center of the parallelepiped, and hence is representative of the domain. The domains $[\mathbf{u}_1]$ and $[\mathbf{u}_2]$ are then updated accordingly translating $[\tilde{\mathbf{u}}_1]$ and $[\tilde{\mathbf{u}}_2]$, i.e. $[\mathbf{u}_k] = [\tilde{\mathbf{u}}_k] + C^{-1}(\tilde{\mathbf{x}} - \tilde{\mathbf{x}}_k)$.

5 Experiments

Experiments presented in this section cover a wide range of situations, from 2D implicit functions to higher dimensional systems. For comparing the usage of parallelepiped domains and box domains, the volume of each enclosure will be compared. In order to have a dimension free measure, the *reduced-volume*, defined as the n^{th} root of the volume where n is the dimension of the problem, will be used. Note that the volume of a parallelepiped is simply the volume of its characteristic domain multiplied by the absolute value of the determinant of its characteristic matrix. The algorithms have been run on a Intel(R) Core(TM)2 Duo CPU at 2.20 GHz, with 4Gb of memory, under Windows XP.

5.1 Intersection of Surfaces

The validated intersection between a sphere and a cylinder is modeled by a CSP with 3 variables and 2 constraints. Both parallelepiped domains and box domains have been used to compute the enclosure of this geometrical object for comparison purpose. Figure 5 shows that, for the same number of bisections, the parallelepiped domains provide a much more accurate enclosure of the solution set. Furthermore, the solution set has been proved rigorously to cross each parallelepiped domain. That information is unavailable when using box domains.

5.2 Two Dimensional Implicit Plot

The problem consists in determining the implicit graph of a complicated function proposed in [1]:

$$f(\mathbf{x}) = x_1 \cos(x_2) \cos(x_1 x_2) + x_2 \cos(x_1) \cos(x_1 x_2) + x_1 x_2 \cos(x_1) \cos(x_2). \quad (11)$$

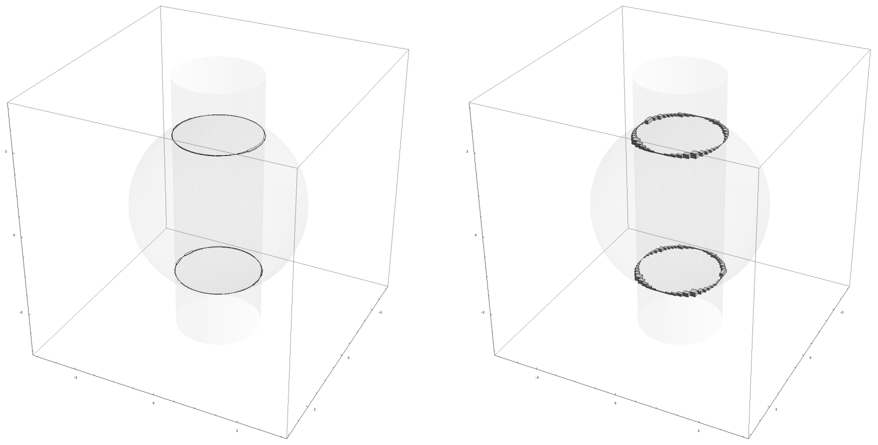

Fig. 5. Intersection of a sphere and a cylinder (both plotted in transparent for information). Enclosures obtained using parallelepiped domains (left) or with box domains (right) after 100 bisections.

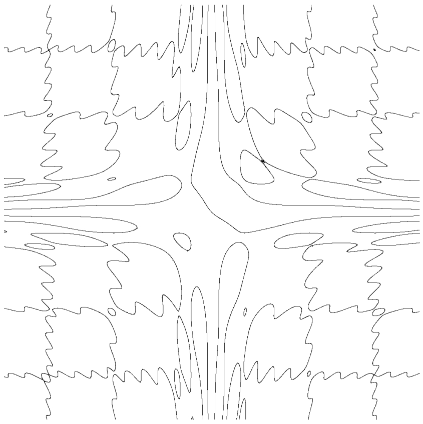

Fig. 6. Verified implicit plot of Tupper's function

Note that in [1] each plus sign is actually consider as a plus/minus sign, hence leading to four different functions, while here we treat (11). After 72 seconds, the enclosure shown on Figure 6 has been computed for $f(\mathbf{x}) = 0$. All the parallelepiped domains are rigorously proved to be crossed by the solution set, this latter being thus completely determined. Timings cannot be compared wrt. the algorithm proposed in [1] because the present approach provides much more information than [1], where a simple outer approximation is computed, exactly as accurate as the pixel size of the resolution.

5.3 The Layne-Watson Exponential Cosine Curve

This system of 3 variables and 2 equations is taken from [2]. This NCSP is an intersection of surfaces whose equations involve compositions of cosine and exponential functions. Once more, the parallelepiped domains provide a more accurate enclosure than box domains, and allow proving the existence of the

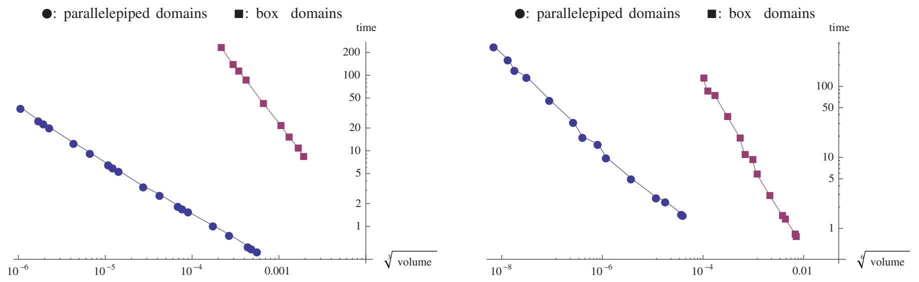

Fig. 7. Log/log plots of the reduced-volume of the enclosure against the time needed to compute this enclosure. Left: The Layne-Watson Exponential Cosine Curve. Right: The Parametrized Broyden Tridiagonal.

whole manifold of solutions. For a more accurate comparison of performances, the left hand side graphic of Figure 7 displays the time needed to obtain a given reduced-volume for both methods. At a first glance, the parallelepiped domains are much more efficient than box domains. Interpreting more precisely these log/log plots, both curves are almost lines. Therefore, both algorithms display a time increasing polynomially with the inverse of the reduced-volume. What is noteworthy is that the slopes of the two lines are different (corresponding to polynomials of different degree), which means that the parallelepiped domains improve not only the timings but also the complexity of the branch and prune algorithm.

5.4 The Parametrized Broyden Tridiagonal

The Broyden tridiagonal problem is a well-known system of n equations and n unknowns [17]. The problem is changed to a parametric problem adding a variable α:

$$(\alpha - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1 = 0 \tag{12}$$

for $1 \leq i \leq n$, with $x_0 = x_{n+1} = 0$ for compact notations. The original value of α is 3, so we choose the domain $[2, 4]$ for this additional variable. As in the original problem, the domains of the other variables are $[-100, 100]$. Introducing such a parameter allows studying how the solutions of the original Broyden tridiagonal problem change with the variations of this parameter. Once more, the usage of parallelepiped domains drastically reduces the time needed to obtain an enclosure of a given reduced-volume, cf. Figure 7. And again, Figure 7 shows that the time needed is approximately a polynomial of the inverse of the reduced-volume, while the usage of parallelepiped domains has reduced the degree of this polynomial.

6 Discussion

These first experiments show that indeed the global constraint contraction performed on parallelepiped domains drastically improves the resolution process (the degree of the polynomial time complexity to reach a given accuracy seems to have been reduced, leading to timings divided by more than 100). Furthermore, the proof of existence of solutions works well for non singular solutions, and allows giving a complete description of the solution set under the form of a sharp enclosure which is proved to contain solutions.

Under-constrained systems of equations appear in many contexts. An important part of the forthcoming work will be to include this framework in an efficient solver to be able to tackle real life problems, like robot workspace computation. Furthermore, parallelepiped domains may improve the efficiency of global optimization algorithms, which often have to solve under-constrained systems of equations. On a theoretical point of view, the introduction of universally quantified parameters in this framework will allow tackling interesting problems, like the robust intersection of surfaces with uncertain parameters.

Acknowledgments. This work was partially funded by the ANR grant number PSIROB06_174445. The authors are grateful to Arnold Neumaier for the useful discussions they had about the relationship between the preconditioned interval Newton operator and global constraints.

References

1. Tupper, J.: Reliable two-dimensional graphing methods for mathematical formulae with two free variables. In: SIGGRAPH 2001, pp. 77–86. ACM, New York (2001)
2. Kearfott, R.B., Xing, Z.: An Interval Step Control for Continuation Methods. SIAM J. Numer. Anal. 31(3), 892–914 (1994)
3. Hansen, E.: Global Optimization Using Interval Analysis, 2nd edn. Marcel Dekker, NY (1992)
4. Merlet, J.: Parallel robots. Kluwer, Dordrecht (2000)
5. Van Hentenryck, P., McAllester, D., Kapur, D.: Solving polynomial systems using a branch and prune approach. SIAM J. Numer. Anal. 34(2), 797–827 (1997)
6. Granvilliers, L., Benhamou, F.: RealPaver: An Interval Solver using Constraint Satisfaction Techniques. ACM Trans. Math. Soft. 32(1), 138–156 (2006)
7. Neumaier, A.: Interval Methods for Systems of Equations. Cambridge U. P, Cambridge (1990)
8. Nedialkov, N.S., Jackson, K.R., Corliss, G.F.: Validated Solutions of Initial Value Problems for Ordinary Differential Equations. Applied Mathematics and Computation 105(1), 21–68 (1999)
9. Neumaier, A.: Overestimation in linear interval equations. SIAM J. Numer. Anal. 24(1), 207–214 (1987)
10. Goldsztejn, A.: A Right-Preconditioning Process for the Formal-Algebraic Approach to Inner and Outer Estimation of AE-solution Sets. Reliab. Comp. 11(6), 443–478 (2005)
11. Goldsztejn, A., Hayes, W.: Reliable Inner Approximation of the Solution Set to Initial Value Problems with Uncertain Initial Value. In: Proc. of SCAN 2006 (2006)
12. Moore, R.: Interval Analysis. Prentice-Hall, Englewood Cliffs (1966)
13. Benhamou, F., Older, W.: Applying Interval Arithmetic to Real, Integer and Boolean Constraints. Journal of Logic Programming 32(1), 1–24 (1997)
14. Benhamou, F., McAllister, D., Hentenryck, P.V.: CLP(Intervals) Revisited. In: International Symposium on Logic Programming, pp. 124–138 (1994)
15. Collavizza, H., Delobel, F., Rueher, M.: Comparing Partial Consistencies. Reliab. Comp. 1, 1–16 (1999)
16. Goldsztejn, A.: A Branch and Prune Algorithm for the Approximation of Non-Linear AE-Solution Sets. In: Proc. of ACM SAC 2006, pp. 1650–1654 (2006)
17. Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Software 7(1), 136–140 (1981)