Reducing pessimism in Interval Analysis using Bsplines Properties: Application to Robotics

Sebastien Lengagne, Rawan Kalawoun, François Bouchon, Youcef Mezouar

Reducing pessimism in Interval Analysis using Bsplines Properties: Application to Robotics

Sébastien Lengagne,
Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France.
sebastien.lengagne@uca.fr

Rawan Kalawoun*
Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France.
rawan.kalawoun@uca.fr

François Bouchon
CNRS UMR 6620 and Clermont-Auvergne University, Mathématiques Blaise Pascal, Campus des Cézeaux, 3 place Vasarely, TSA 60026, CS 60026, 63178 Aubière Cedex France
francois.bouchon@uca.fr

Youcef Mezouar
Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France.
youcef.mezouar@uca.fr

Abstract

Interval Analysis is interesting to solve optimization and constraint satisfaction problems. It makes possible to ensure the lack of the solution or the global optimal solution taking into account some uncertainties. However, it suffers from an over-estimation of the function called pessimism. In this paper, we propose to take part of the BSplines properties and of the Kronecker product to have a less pessimistic evaluation of mathematical functions. We prove that this method reduces the pessimism, hence the number of iterations when solving optimization or constraint satisfaction problems. We assess the effectiveness of our method on planar robots with 2-to-9 degrees of freedom and to 3D-robots with 4 and 6 degrees of freedom.

*R. Kalawoun was supported by the French Government through the FUI Program (20th call with the project AEROSTRIP).

Keywords: Interval Analysis, Bsplines, Kronecker product, pessimism, robotics optimization

1 Introduction

In robotics, lots of applications require getting optimal parameters. For instance, during identification, the optimization process makes possible to find parameters matching the results of the model with the experimental results [1]. Optimization techniques allow robots to perform motions or static pose with minimal energy consumption or to perform tasks as fast as possible [2]. Those techniques provide optimal parameters that must satisfy a set of constraints (such as joint limits, joints torques, balance, ...) in order to ensure the integrity of the robotic system and the task completion and optimize a given criteria. Depending on the application the constraints and criteria function can vary between simple (continuous, linear, monotonic, ...) and very complex (discontinuous, non-linear, with local extremum ...).

In simple cases (quadratic criteria without constraint), the optimal parameters can be found easily using Lagrange multipliers, active set [3], or some other methods. For more complex cases, one usually refer to heuristic algorithms or to some iterative algorithms [4, 5] that produce a solution in a finite time. For relatively complex cases (especially with discontinuous or non differentiable functions), Genetic Algortihms return a solution without any guarantee of optimality.

Optimization techniques based on Interval Analysis provide global optimal results ensuring constraints satisfaction (in a finite space) and can ensure the lack of solutions in case of infeasible problem [6]. Those algorithms start from the initial search space and iteratively split the search space try to find a set of solution that fit the problem. Unfortunately those techniques suffer from a prohibitive computation time, due to a large number of iterations caused by the pessimism induced by the use of Interval Analysis. The pessimism is an over-estimation of the functions, i.e. the given interval is still conservative (it contains the actual solution) but it is larger than the actual solution. Thus, the pessimism may increase the number of iterations of the optimization algorithms. Pessimism is deeply linked to the inclusion function (the way to evaluate a function in Interval Analysis). In order to reduce pessimism, the natural inclusion function can be replaced by the centred or the Taylor-centred inclusion [7].

In this paper, we propose to use the BSplines and the Kronecker product properties to evaluate the functions in a less conservative way in order to decrease the pessimism, hence to decrease the number of iterations and the computation time of optimization processes. This method was already applied in a one dimension case to evaluate functions over time intervals [2]. In this paper, we present the extension to multi-dimensional problems, how to use it during optimization problem resolution. We assess the results of our method on robotics systems.

We present the optimization problem, how to solve it using Interval Analysis and we explain the impact of the pessimism on the number of iterations in Section 2. In Section 3, we develop our method based on the Bsplines and Kronecker product properties that reduces the pessimism. We assess our method on planar and 3D robot systems in Section 4. Eventually, we discuss the proposed method in Section 5.

2 Optimization using Interval Analysis

2.1 Problem statement

In this paper, we present a new way to evaluate function over multi-dimensional interval. We assess our method through the resolution of optimization problems such as:

$$\begin{array}{rl} \text{Find} & q \in \mathbb{Q} \subset \mathbb{R}^n \\ \text{such as} & \min\limits_{q} \mathcal{F}(q) \\ \text{with } \forall j \in \{1,\ldots,m\} & \mathcal{G}_j(q) \in [\underline{g}_j, \overline{g}_j] \end{array} \tag{1}$$

Where n is the number of parameters, $\mathbb{Q}$ the finite search space, $\mathcal{F}(q)$ the criteria function and $\mathcal{G}(q)$ the set of j constraints that must remain within the interval $[\underline{g}_j, \overline{g}_j]$ with the lower bound $\underline{g}_j$ and the upper bound $\overline{g}_j$ limits.

This problem may be turned into its interval analysis form, such :

$$\begin{array}{rl} \text{Find} & [q] \in \mathbb{IQ} \subset \mathbb{IR}^n \\ \text{such as} & \min\limits_{[q]} \mathcal{F}([q]) \\ \text{with } \forall j \in \{1,\ldots,m\} & \mathcal{G}_j([q]) \in [\underline{g}_j, \overline{g}_j] \end{array} \tag{2}$$

With $\mathbb{IR}$ and $\mathbb{IQ}$ represent the set of all the possible interval in $\mathbb{R}$ and $\mathbb{Q}$.

2.2 Interval Analysis

Interval Analysis (IA) was initially developed to take into account the quantification errors introduced by the floating point representation of real numbers with computers [8, 9, 10]. IA methods have been later used to solve optimization problems. Several works showed that IA is competitive compared to the classic optimization solvers since it provides guaranteed solutions respecting the constraints [11, 12, 13]. Nowadays, IA is largely used in robotics [14, 15]. Recent works use IA to compute robot trajectories and the guaranteed explored zone by a robot [16, 17, 18].

Let us define an interval $[a] = [\underline{a}, \bar{a}]$ as a connected and closed subset of $\mathbb{R}$, with $\underline{a} = Inf([a])$, $\bar{a} = Sup([a])$ and $Mid([a]) = \frac{\underline{a}+\bar{a}}{2}$. The set of all real intervals of $\mathbb{R}$ is denoted by $\mathbb{IR}$. A vector of interval is defined as a box. Real arithmetic operations are extended to intervals. Consider an operator $\circ \in \{+, -, *, \div\}$ and $[a]$ and $[b]$ two intervals. Then:

$$[a] \circ [b] = [inf_{u\in[a],v\in[b]}\, u \circ v, \quad sup_{u\in[a],v\in[b]}\, u \circ v] \tag{3}$$

Consider a function $\mathbf{m} : \mathbb{R}^n \longmapsto \mathbb{R}^m$; the range of this function over a box [a] is given by:

$$\mathbf{m}([\mathbf{a}]) = \{\mathbf{m}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{4}$$

The interval function $[\mathbf{m}] : \mathbb{IR}^n \longmapsto \mathbb{IR}^m$ is an inclusion function for $\mathbf{m}$ if:

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{m}([\mathbf{a}]) \subseteq [\mathbf{m}]([\mathbf{a}]) \tag{5}$$

The natural inclusion function of $\mathbf{m}$ is evaluated by replacing each occurrence of a real variable by the corresponding interval and each standard function by its interval counterpart.

2.3 Pessimism

Through inclusion functions, we can perform a conservative evaluation of a function on a whole set of values. Since it deals with numerical truncation, imprecise inputs and round-off errors, Interval methods are reliable. A lack of accuracy is possible in IA, when the overestimation produced by the underlying inclusion function is large [19]. However, accuracy is required in optimization methods as presented in Section 2.4.

This overestimation of the actual bounds of the results is called the pessimism. It is mainly caused by the multi-occurrence of variables in equations [20, 21]. For instance, consider the equation $y = (x+1)^2$ with $x \in [-1, 1]$. Using the latter formulation, we find $y = [0, 4]$, but using the following expression $y = x^2 + 2x + 1$, the result is $y = [-1, 4]$. Both results are correct, but the solution range is larger using the second expression due to the pessimism. The pessimism is reduced by designing smart inclusion functions in several works [22, 23, 24].

2.4 Solving optimization problem : Bisection

To prove that our method is effective, we will compare the performances (ie. number of iterations and computation time) of a basic optimization algorithm using the natural inclusion function and our BSplines inclusions function. Bisection algorithms, such as presented in Algorithm 1 allow to solve optimization problem defined by Equation 2 using Interval Analysis. The Bisection method consists in evaluating the constraints and the criteria function for a current box. Those evaluations are used to determine if this box must be through away (due to one constraint violation or too large criteria function) or split into two smaller boxes. The splitting process aims at getting a tighter evaluation of the constraints and of the function. The bisection process ends when the size of the box is smaller than a given threshold. The constraint $\mathcal{G}_j([q])$ is considered as violated if its interval evaluation as no intersection with the bounds $[g_j]$.

Several splitting method can be applied [9]. For sake of simplicity, (since in this paper we focus on the evaluation function), we split the box into two boxes by splitting the largest interval in the middle.

2.5 Contraction

Contraction is based on the filtering algorithm concept [25]. It manipulates the equation in order to propagate the constraints in two ways: from inputs to outputs (Eq.(6)) and from outputs to inputs (Eq.(7)): this is the classic forward/backward contractor:

$$\forall j \quad [g_j] \leftarrow \mathcal{G}_j([q]) \cap [g_j] \tag{6}$$

$$\forall i, j \quad [q_i] \leftarrow \mathcal{G}_j^{-i}([q], [g_j]) \cap [q_i] \tag{7}$$

where $[g_j]$ is the current value of the constraints bounds (initialized to $[\underline{g}_j, \overline{g}_j]$), q_i the i-th contractor input, $[q_i]$ the current bounds on the input q_i and $\mathcal{G}_j^{-i}(q, [g_j])$ is the inverse of function $\mathcal{G}_j(q)$ regarding the input q_i taking into account the current value of the constraints bounds $[g_j]$. $[q_i]$ and $[g_j]$ can be different from the initial bounds due to previous iterations of the algorithm. Eventually, contraction may define a smaller subset of input boxes respecting the different constraints.

For instance, given three variables $a \in [-\infty, 5]$, $b \in [-\infty, 4]$ and $c \in [6, \infty]$, and the constraint $c = a + b$, find the intervals of a, b, c respecting this constraint. One can

Algorithm 1 Algorithm for optimization process using bisection process.

Require: Initial research space $\mathbf{Q}$, desired precision ε
  initialization : Q.push($\mathbf{Q}$), $\tilde{f} = \infty$
  while Q is not empty **do**
    get element $[q] = Q$.pop()
    evaluate $[f] = [\underline{f}; \overline{f}] = \mathcal{F}([q])$
    if $\underline{f} < \tilde{f}$ **then**
      evaluate constraints: $\forall j\ [g_j] = \mathcal{G}_j([q])$
      if $\forall j\ [g_j] \cap [\underline{g}_j; \overline{g}_j] \neq \emptyset$ **then**
        if $\forall j\ [g_j] \cap [\underline{g}_j; \overline{g}_j] = [g_j]$ **and** $\overline{f} < \tilde{f}$ **then**
          $\tilde{f} = \overline{f}$
          $\tilde{q} = q$
        end if
        if $diam([q]) > \varepsilon$ **then**
          $\{q_1, q_2\} = bisection(q)$
          Q.push(q_1)
          Q.push(q_2)
        end if
      end if
    end if
  end while
  return Optimal box $\tilde{q}$ of problem of Eq.(2)

process as follow:

$$\begin{array}{lll} c = a + b & \Rightarrow c \leftarrow [c \cap (a+b)] & \Rightarrow c = [6, 9] \\ a = c - b & \Rightarrow a \leftarrow [a \cap (c-b)] & \Rightarrow a = [2, 5] \\ b = c - a & \Rightarrow b \leftarrow [b \cap (c-a)] & \Rightarrow b = [1, 4] \end{array} \tag{8}$$

Thus, by using contractors, the intervals of the variables become tighter: $a \in [2, 5]$, $b \in [1, 4]$ and $c \in [6, 9]$.

Generally, a combination of contractions and bisections is used. In Algorithm 1, the contraction step is done just before the evaluation step. However, this method may still suffer from pessimism explained in Section 2.3. In Section 3, we propose a new method to decrease pessimism.

3 Function evaluation and pessimism.

As presented before, the pessimism is linked to the mathematical expression of the inclusion function. To assess the effectiveness of our method, in this paper, we consider generic natural inclusion functions and we use the Bsplines and Kronecker product properties to have a less pessimistic evaluation.

3.1 BSplines properties

BSplines function is the weighted sum of several basis functions. For a one dimensional case $q \in \mathbb{R}$, a $k-$ order BSplines function is defined by m control points p_i and basis functions $b_i^k(q)$ k such as:

$$F(q) = \sum_{i=1}^{m} b_i^k(q) p_i \tag{9}$$

A BSplines curve is inside the convex hull of its control poly-line [26]. This property is obtained quite easily from the following definition of a basis function:

$$\forall q \in [\underline{q}, \overline{q}] \quad \sum_{i=1}^{m} b_i^k(q) = 1 \tag{10}$$

This immediately yields:

$$\forall i \in [1, m] \quad \underline{F} \leq p_i \leq \overline{F} \Rightarrow \forall q \in [\underline{q}, \overline{q}] \quad \underline{F} \leq F(q) \leq \overline{F} \tag{11}$$

A conservative estimation of the bounds of $F(q)$ is made based on the minimum and the maximum of the control points.

Let us reformulate Equation (9) by assuming uniform Bsplines (i.e. $m = k + 1$):

$$F(q) = \mathbf{B_K} \times [p_1, p_2, \ldots, p_m] \times [1, q, q^2, \ldots, q^k]^T \tag{12}$$

where $\mathbf{B_K} \in \mathbb{R}^{m \times m}$ is the matrix containing the polynomial coefficients of the basis functions $b_i^K(q)$. Hence, knowing the polynomial formulation of $F(q)$

$$F(q) = [a_1, a_2, \ldots, a_k] \times [1, q, q^2, \ldots, q^k]^T \tag{13}$$

One can easily deduce the link between the equivalent control points and the polynomial coefficients:

$$[a_1, a_2, \ldots, a_k] = \mathbf{B_K} \times [p_1, p_2, \ldots, p_m] \tag{14}$$

Eventually, one can compute the equivalent control points through :

$$[p_1, p_2, \ldots, p_m] = \mathbf{B_K}^{-1} \times [a_1, a_2, \ldots, a_k] \tag{15}$$

This properties was already used to tackle pessimism in one dimension [2]. In this paper, we propose a generalization of this concept to multi-dimensional problems using the Kronecker product formulation.

3.2 Kronecker Product Formulation

As presented in [27], for n-dimensional case $q \in \mathbb{R}^n$, the Bsplines functions can be written as:

$$F(q) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \ldots \sum_{z=1}^{m_n} \left(b_i^{m_1}(q_1) b_j^{m_2}(q_2) \ldots b_z^{m_n}(q_n) \right) \times p_{i,j,\ldots,z} \tag{16}$$

The previous equation can be written though its polynomial form:

$$F(q) = [a_1, a_2, a_3, \ldots, a_i, \ldots] \times [1, q_1, q_2, q_1 q_2, \ldots, \mu_i, \ldots]^T \tag{17}$$

where μ_i represents the i-th monomial. By setting P the vector of all the control points $p_{i,j,\ldots,z}$ and X the vector of all the monomial coefficients, we generalize Equation (15) to multi-dimensional case:

$$P = \mathbb{B}^{-1} X \tag{18}$$

where $\mathbb{B}$ can be written as:

$$\mathbb{B} = \mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \ldots \otimes \mathbf{B}_i \otimes \ldots \otimes \mathbf{B}_n \tag{19}$$

The matrix $\mathbf{B}_i$ links the control points to the coefficient of the polynomial expression of the basis functions of input q_i and $\otimes$ is the Kronecker product. The Kronecker product was firstly studied in the nineteenth century [28]. The Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{n_1,m_1}$ and $\mathbf{B} \in \mathbb{R}^{n_2,m_2}$ is the matrix $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{n_1 \times n_2, m_1 \times m_2}$ defined as follows:

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & \ldots & a_{1,n_1} \\ \vdots & \vdots & \vdots \\ a_{m_1,n_1} & \ldots & a_{m_1,n_1} \end{pmatrix} \tag{20}$$

$$\mathbf{A} \otimes \mathbf{B} = \left(\begin{array}{c|c|c|c} a_{1,1} \times \mathbf{B} & a_{1,2} \times \mathbf{B} & \ldots & a_{1,n_1} \times \mathbf{B} \\ \hline a_{2,1} \times \mathbf{B} & a_{2,2} \times \mathbf{B} & \ldots & a_{2,n_1} \times \mathbf{B} \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline a_{m_1,1} \times \mathbf{B} & a_{m_1,2} \times \mathbf{B} & \ldots & a_{m_1,n_1} \times \mathbf{B} \end{array}\right) \tag{21}$$

More properties of the Kronecker product can be found in [29]. The inversion of the Kronecker product is given by:

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1} \tag{22}$$

Using Equation (22), Equation (18) can be turned into :

$$P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} \otimes \ldots \otimes \mathbf{B}_n^{-1}\right) X \tag{23}$$

Assuming $\mathbb{B} \in \mathbb{R}^{x,x}$, with $x = \prod_{j=1}^{n} m_j$, the complexity of the inversion of the matrix $\mathbb{B}$ in Equation (18) is $\mathcal{O}(x^3)$. Using the invertible property presented in Equation (22), the complexity decreases to $\mathcal{O}(x \sum_n m_j^2)$, that will make possible to test some case, especially for large n, where inverting directly $\mathbb{B}$ with the $\mathcal{O}(x^3)$ complexity is prohibitive regarding the computation time.

3.3 Inputs normalization

In Equation (18), the authors of [27] consider that the vector remains constant over whole the computation process and update the BSplines matrix $\mathbf{B}_i$ regarding the current bounds of $[q_i]$ at each iterations. The main drawback of this solution is the numerical errors while computing ${\mathbf{B}_i}^{-1}$ for narrow intervals on $[q_i]$ due to ill conditioning. To solve this issue, we use the inputs normalization with the following affine change of variables:

$$[q_i] = m_i + [q_i^{ref}]\frac{d_i}{2} \tag{24}$$

with m_i and d_i are the middle and the diameter of interval $[q_i]$ and $[q_i^{ref}]$ is set as the reference interval of $[q_i]$ and is initialized at $[q_i^{ref}] = [-1; 1]$. By doing so, the Bsplines matrix $\mathbb{B}_i$ rely on the reference interval $[q_i^{ref}]$, hence remains constant over whole the computation process. The polynomial parameter vector X relies on the middle and diameter of all the inputs.

3.4 Non linear functions

The proposed method in this paper is based on a polynomial formulation of the constraint and criteria functions. In case of non-linear functions $s([q_i])$, we use a Taylor approximation:

$$s([q_i]) = s(m_i) + \frac{ds(m_i)}{dq}\frac{d_i}{2}[q_i^{ref}] + \ldots + [\varepsilon_s(q_i)]) \tag{25}$$

Where $[\varepsilon_s(q_i)]$ is normalized using $[\varepsilon_{s,i}{}^{ref}]$, $m_{s,i}$ and $d_{s,i}$ as proposed in Equation (24). The reference error interval $[\varepsilon_{s_i}{}^{ref}]$ is considered as a new input in Equation (19). The interval value $[\varepsilon_s(q_i)]$, hence $m_{s,i}$ and $d_{s,i}$, is updated as soon as the original interval $[q_i]$ is modified and $[\varepsilon_{s,i}{}^{ref}]$ is set to $[-1; 1]$. $[\varepsilon_s(q_i)]$ is not considered in the bisection process. In our work, we consider the first order approximation of non-linear functions. Since, we aim at dealing with robotics (non-linear) problems in Section 4, we consider the following non-linear functions: $s([q_i]) \in \{\sin([q_i]), \cos([q_i])\}$:

3.5 Function Evaluations samples

In this section, we present the procedure to compute the results of a two-dimensional function based on the BSplines and Kronecker product properties. and we compare the results of the different inclusion functions for a 1-dimensional example.

3.5.1 2-dimensional example

As an example of the use of BSplines to reduce pessimism, let us consider

$$f(q_1, q_2) = 1 - 3q_1 - 2q_2 + 4q_1q_2 \tag{26}$$

with $q_1, q_2 \in [-10, 10]$. Thus, we have $X = [1, -3, -2, 4]^T$. The details for the original method (without input normalization) can be found in the appendix of [30]. After normalization of the inputs q_1 and q_2 using those equations $q_1 = 10q_1^{ref}$ and $q_2 = 10q_2^{ref}$, we can deduce that :

$$f(q_1^{ref}, q_2^{ref}) = 1 - 30q_1^{ref} - 20q_2^{ref} + 400q_1^{ref}q_2^{ref} \tag{27}$$

Thus, we have $X = [1, -30, -20, 400]^T$ and we compute $\mathbb{B}$. Since f contains two variables q_1 and q_2 each one of degree $n = 1$ the basis function for each variable is :

$$\begin{array}{l} b_1^1(q) = 0.5 - 0.5q \\ b_2^1(q) = 0.5 + 0.5q \end{array} \tag{28}$$

Thus, we found $\mathbf{B}_1 = \mathbf{B}_2 = \begin{bmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \end{bmatrix}$ and we compute Hence, $\mathbf{B}_1^{-1} = \mathbf{B}_2^{-1} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$ and get the matrix for the function f :

$$\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} = \begin{pmatrix} 1 \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} & -1 \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \\ 1 \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} & -1 \times \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \end{pmatrix} = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{29}$$

We can compute the equivalent control point using:

$$P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1}\right) X = [451, -409, -389, 351]^T \tag{30}$$

Thus, using our method, we can deduce that $f \in [-409, 451]$ which is the same interval obtained without inputs normalization. By applying Horner schemes, we got two different expressions of f: $f = 1 + q_1(-3 + 4q_2) - 2q_2$ and $f = 1 - 3q_1 - 2q_2(1 - 2q_1)$. For those two different expressions we got two intervals $[-449, 451]$ and $[-449, 451]$. Once can deduce that our method reduces pessimism in this example. Horner scheme works for $n = 2$ and gives the same results, but his behaviour becomes more critical when n increases. Hence, the proposed method allows to avoid or at least reduce pessimism for $n = 2$. Moreover, if the value of n increases, then the pessimism will be more reduced: the multi-occurrence increases once n increases.

Now, let consider an other for value for $q_1 = q_2 \in [-1, 1]$ for the same Equation 26. Thus, we have $X = [1, -3, -2, 4]^T$. We can compute the equivalent control point using: $P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1}\right) X = [10, -4, -2, 0]^T$ Thus, using our method, we can deduce that $f \in [-4, 10]$. However, using the natural inclusion functions we obtain $f \in [-8, 10]$. And using the two different Horner schemes already defined we can deduce that $f \in [-8, 10]$. Hence, we can deduce that our method reduces pessimism for large and tight intervals.

3.5.2 1-dimensional example

We also validate our method on a non linear function f defined by

$$f(x) = x^2 + sin(x) \tag{31}$$

and the intervals $[x_1] = [\frac{2\pi}{3}, \frac{4\pi}{3}]$ and $[x_2] = [\frac{99\pi}{100}, \frac{101\pi}{100}]$. We will compare the approximations of $f(x_1)$ and $f(x_2)$ obtained by using the natural $[f]_n$, the centred $[f]_c$, Taylor of order two $[f]_{T2}$, Chebyshev of order five $[f]_{C5}$,, Bernstein inclusion function $[f]_B$, BSplines and minimal inclusion functions $[f]_{Bs}$ and the actual (without pessimism) evaluation $[f^*]$ respectively.

$$\begin{aligned}
&[f]_n([x]) = [x]^2 + sin([x])\\
&[f]_c([x]) = f(\pi) + ([x] - \pi)[f']([x])\\
&[f]_{T2}([x]) = f(\pi) + ([x] - \pi)[f']([\pi]) + \frac{([x] - \pi)^2}{2}[f'']([x])\\
&[f]_{C5}([x]) = \frac{1}{2}f_0 + [-1, 1]\sum_{i=1}^{5}|f_i|\\
&[f]^*([x]) = [\underline{x}^2 + sin(\underline{x}), \bar{x}^2 + sin(\bar{x})]
\end{aligned}$$

with $f'(x) = 2x + cos(x)$ and $f''(x) = 2 - sin(x)$ and $f_i = \frac{2}{\pi}\int_0^\pi f(cos(x))cos(ix)dx$. The results are presented in Table 1 where $\triangle\,([f]([x]))$ represents the different between the width of the computed interval and the actual width of the interval such as

$$\triangle\,([f]([x])) = diam([f]([x])) - diam([f]^*([x])).$$

As we can see, the natural inclusion function remains competitive for the large intervals. However, the centred, Taylor and Chebyshev inclusion functions are more

	$[x_1] = [\frac{2\pi}{3}, \frac{4\pi}{3}]$		$[x_2] = [\frac{99\pi}{100}, \frac{101\pi}{100}]$	
$[f]$	$[f]([x_1])$	$\triangle$ $([f]([x_1]))$	$[f]([x_2])$	$\triangle$ $([f]([x_2]))$
$[f]_n$	$[3.52046, 18.41199]$	3.46410	$[9.64178, 10.09940]$	0.12564
$[f]_c$	$[1.62022, 18.11899]$	5.07134	$[9.70163, 10.03758]$	0.00397
$[f]_{T2}$	$[4.33706, 16.97362]$	1.20913	$[9.70362, 10.03659]$	0.00099
$[f]_{C5}$	$[4.15463, 16.68117]$	1.09911	$[9.70362, 10.03657]$	0.00098
$[f]_B$	$[5.25251, 16.67994]$	0	$[9.70461, 10.036673]$	0.000083
$[f]_{Bs}$	$[5.25251, 16.67994]$	0	$[9.70461, 10.0366]$	0.00002
$[f]^*$	$[5.25251, 16.67994]$	0	$[9.70461, 10.03658]$	0

Table 1: Comparing inclusion functions

efficient than the natural inclusion function for the small intervals. Besides, Taylor and Chebyshev inclusion functions bring noticeable improvement compared to the natural and the centred inclusion functions even for large intervals. Moreover, the best two inclusion functions are Bernstein expansion and Bsplines inclusion functions. However, it is clear, on this example, that Bsplines inclusion function remains the best inclusion function for large and small intervals: the results are almost the actual intervals.

3.6 Constraint Evaluation

In this part, we focus on the computation of the bounds of a function $g_j([q])$ and, especially, on an efficient determination of the constraint violation or satisfaction regarding a given bound interval $[g_j]$. We recall that the evaluation of the constraint function $[g_j]([q])$ is done by computing the minimum and maximum value of the vector P such as computed in Equation 18. The size of matrix $\mathbb{B}$ and vector X may be large depending of the number of variables and of the order of each variables. The multiplication of Equation 18 may imply a large number of operations, hence a huge computation time that can be reduced as presented in this section.

As presented in Algorithm 1, this evaluation is used in order to assess if the corresponding constraint remains within a given bounds $[\underline{g}_j, \overline{g}_j]$. By sequentially computing the control point P_k, we can determine if the corresponding box is overlapping when $\min\limits_{\forall k} P_k < \underline{g}_j$ and $\max\limits_{\forall k} P_k > \underline{g}_j$ or $\min\limits_{\forall k} P_k < \overline{g}_j$ and $\max\limits_{\forall k} P_k > \overline{g}_j$. As soon as, the overlapping of the constraint is detected, there is no need to compute the other control points, since the bisection must be performed, hence the evaluation of the process is stopped: there may be no need to compute all the control points P_k. First, let us decompose Equation (18) as follow:

$$P = \mathbb{B}^{-1} X^I + \mathbb{B}^{-1} X^\varepsilon \tag{32}$$

where $P \in \mathbb{R}^x$ is the set of the equivalent control points, $X^I \in \mathbb{R}^x$ contains the polynomial coefficients that refer only to monomials with reference intervals ($[q_i^{ref}]$) and $X^\varepsilon \in \mathbb{R}^x$ contains the polynomial coefficients that refer to monomial with reference intervals ($[q_i^{ref}]$) and reference approximation errors ($[\varepsilon_{s_i}^{ref}]$). Obviously $X = X^I + X^\varepsilon$.

Taking into account that the vectors X^I and X^ε contains a lot of null values, it is computationally relevant to use a sparse representation of the matrices and vectors

such as:

$$P = \mathbb{S}^I X_s^I + \mathbb{S}^\varepsilon X_s^\varepsilon \tag{33}$$

where $X_s^I \in \mathbb{R}^y$ is the vector collecting the non-null coefficients of X^I and $\mathbb{S}^I \in \mathbb{R}^{x,y}$ is obtained by removing the columns corresponding to the null coefficient of X^I from $\mathbb{B}^{-1}$, and similarly for $X_s^\varepsilon \in \mathbb{R}^z$ and $\mathbb{S}^\varepsilon \in \mathbb{R}^{x,z}$.

Since the second term of Equation (33) $\mathbb{S}^\varepsilon X_s^\varepsilon$ relies on non-linear approximation errors, it must decrease as the width of the input boxes decreases. In order to speed up the computation time, we introduce the non-linear approximation error interval $[\epsilon] \in \mathbb{IR}^n$ through:

$$[\epsilon] = \mathbf{S}^\varepsilon . X_s^\varepsilon \tag{34}$$

where $\mathbf{S}^\varepsilon \in \mathbb{IR}^z$ is an interval vector where each element is the interval union of all the values of the corresponding column of $\mathbb{S}^\varepsilon$. Thus, each equivalent control point given by $\mathbb{S}^\varepsilon X_s^\varepsilon$ lies in the interval $[\epsilon]$. Obviously, $[\epsilon]$ is a pessimistic evaluation of the non linear approximation errors, but it is much faster to compute the dot product of the $z-$dimensional vector $\mathbf{S}^\varepsilon . X_s^\varepsilon$ than using the the product of $x \times z$-matrix with the $z-$dimensional vector. Eventually, the evaluation of the constraint $g_j(q)$ is done from:

$$P = \mathbb{S}^I . X_s^I + [\epsilon] \tag{35}$$

Hence, the function can be evaluated throught the minimal and maximal value of P, where each control point P_k can be computed as:

$$P_k = {\mathbb{S}^I}_k . {X_s^I}_k + [\epsilon] \tag{36}$$

The polynomial formulation of the constraints and the computation of the sparse matrices and vectors are done once in a preparation phase, before the beginning of Algorithm 1.

3.7 Constraint Contraction

As presented in Section 2.5, the contraction of one constraint consists in updating the bounds of the constraint functions and the bound of the inputs. In order to perform the contraction, we consider a new function $\nu_j([q], [g_j])$ such as:

$$\nu_j([q], [g_j]) = g_j([q]) - [g_j] \tag{37}$$

The control points of this function P_{ν_j}can be evaluated through

$$P_{\nu_j} = \mathbb{B}_{\nu_j}^{-1} X_{\nu_j} \tag{38}$$

where $\mathbb{B}_{\nu_j}^{-1}$ and X_{ν_j} are the matrix and vector representation of the function $\nu_j([q], [g_j])$. We perform the same decomposition than the one presented in Equation (35) to deal with error component and sparsity of the vector X_{ν_j}.

Let us define $[\mu]$ as the interval value to contract that may be the bounds of the constraint $[g_j]$ or the bounds of the reference intervals $[q_i^{ref}]$ and their sine and cosine approximation errors $[\varepsilon_{sin}(q_j)]$ and $[\varepsilon_{cos}(q_j)]$. The contraction process presented in Equation (7) turns to:

$$[\mu] \leftarrow [\mu] \cap (\nu_j([q], [g_j]) + [\mu]) \tag{39}$$

Therefore, the equivalent control point P_μ of $\nu_j([q], [g_j]) + [\mu]$ can be computed as:

$$P_\mu = \mathbb{B}_{\nu_j}^{-1} X_{\nu_j} + \mathbb{B}_{\nu_j}^{-1} X_\mu = P_{\nu_j} + \mathbb{B}_{\nu_j}^{-1} X_\mu \tag{40}$$

where the values of the vector X_μ are zero except for the component corresponding to the monomial μ that is the opposite of the one of X_{ν_j}.

Eventually, the evaluation of Equation (39) is performed as presented in Section 3.6 taking into account the error such as in Equation (35) and using the sparse properties especially because X_μ contains only one non-zero value.

4 Results

The following abbreviations will be used in this section :

- BI: Bisection process using Interval Analysis,
- CI: Contraction and bisection using Interval Analysis,
- BS: Bisection process using the BSplines and Kronecker product properties,
- CS: Contraction and bisection process using the BSplines and Kronecker product properties.

To compare the methods, with the same implementation performance, we re-code the state of the art interval analysis without using available libraries such as ALIAS [31] or IBEX [32]. For sake of simplicity, our implementation do not deal with rounding errors. The sine and cosine functions in state of the art Interval Analysis returns intervals without pessimism. In order to focus on the performance of our methods, the optimization process are provided taking into account a one-thread computation process. Our method was programmed using the C++ language and executed on the following hardware and software: CPU Intel(R) Xeon(R) E5-2670, 6.4 GHz, Cache 8 Mo: CentOS Linux release 7.5.1804 (Core) 64 bits.

4.1 2D Robot feasible space

In this section, we present the computation of the feasible space of a 2-dof planar robot with two links of length 1 unit, as presented in Figure 1, such as:

$$\begin{array}{rl} \text{Find all} & [q] \in \mathbb{Q} \subset \mathbb{R}^n \\ \text{such as } \forall j \in \{1, \ldots, m\} & \mathcal{G}_j([q]) \in [g_j] \end{array} \tag{41}$$

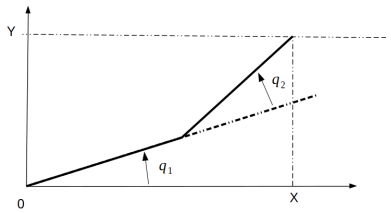


Figure 1: Representation of the 2-dof robot.

Since the resolution of this problem is straightforward from the resolution of the optimization problem, we do not detail it in this paper. The system is a 2-dof planar robot $q \in \mathbf{R}^2$, with joint limits $q_1, q_2 \in [-3; 3]$, we set the following constraints :

$$\begin{array}{rll} x = & \cos(q_1) + \cos(q_1)\cos(q_2) - \sin(q_1)\sin(q_2) & \in [1.2; 1.8] \\ y = & \sin(q_1) + \cos(q_1)\sin(q_2) + \sin(q_1)\cos(q_2) & \in [-0.5; 0.5] \end{array} \tag{42}$$

The results are presented in Figure 2 and in Table 2 with a stopping threshold of 0.05. The feasible boxes are represented in red and the possible boxes are green. For the state-of-the-art methods (BI and CI), Figures 2(a) and 2(c) show that the pessimism produces possible boxes even if they are totally inside or outside the feasible space.

One can see that the use of Bsplines properties decreases the pessimism since the number of possible boxes (green) is reduced and most of them are partially inside and outside the feasible space. CI method reduces the number of iterations regarding BI method as presented in Table 2 even if only a few possible boxes are removed.

Our new BS method allows the number of iterations to decrease. This method produces fewer but larger possible boxes as it is shown on Figure 2(b). Moreover, our new CS method reduces the number of iterations (nearly half of the BI method) and the number of possible boxes. It also produces a set of interval with different sizes (due to the non-uniform contraction) as presented on Figure 2(d).

Regarding the computation time, the BS method has the lowest computation time even if the preparation phase requires a large computation time ($\simeq$ 250ms). However this computation time can be ignored since it is not required to perform it in case of change of the x or y bound value.

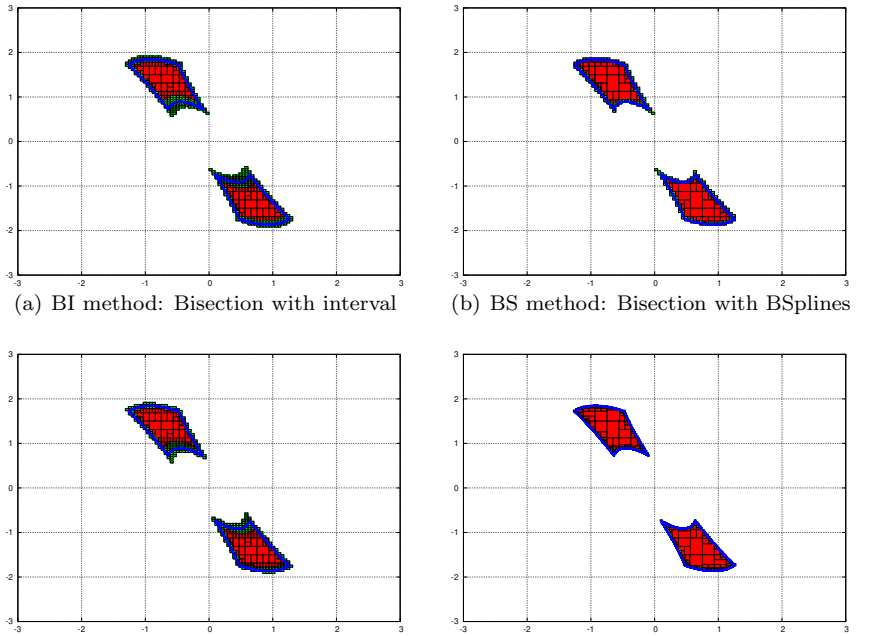
solver	number of iterations	computation time (ms) (+preparation time)	number of feasible boxes	number of possible boxes
BI	1567	24.1	130	716
BS	1083	14 (+242)	102	428
CI	1431	23.7	130	648
CS	743	24 (+262)	136	396

Table 2: Computation results of the feasible spaces for the 2-dof planar robot.

This 2D simple CSP shows that the BSplines inclusion function reduces the pessimism. Hence it reduces the number of iterations, that reduce the computation time for this case. In the following subsection we assess our method on more complex cases dealing with planar 2D-robots and on 3D-robots.

4.2 Planar Robot

We assess our method on multi degrees-of-freedom 2D-robots. We address Constraint Satisfaction Problems (CSP) and Optimization Problems (OP) to find the posture of the robot that fits and optimally fits robot constraints and desired end-effector position. We consider robots with 2-to-9 degrees of freedom, with equal segment length. All the links have the same weight. The Centres Of Mass (COM) are located in the middle of the links. The total length of the robot is equal to 2. We set as a constraint that the COM of the robot must remain in $[-0.1, 0.1]$ on the horizontal plane. We also add a reachability constraint: the effector must reach a specific target. We assess our method on nine different constraint satisfaction problems, with a square target of size 0.2 by 0.2 (pb 1,2,3), a circle target with a radius of 0.1 (pb 4,5,6) and a ring target with an external radius of 0.15 and an interval radius of 0.05 (pb 7,8,9). The targets are localized at three different positions $\{x, y\} : \{0.2; 1.7\}$ (pb 1,4,7),



(a) BI method: Bisection with interval (b) BS method: Bisection with BSplines

(c) CI method: Contraction with interval (d) CS method: Contraction with BSplines

Figure 2: Presentation of the feasible spaces $\{q_1, q_2\}$ for the 2-dof robot. (feasible boxes in red, possible boxes in green, actual feasible space limits in blue.)

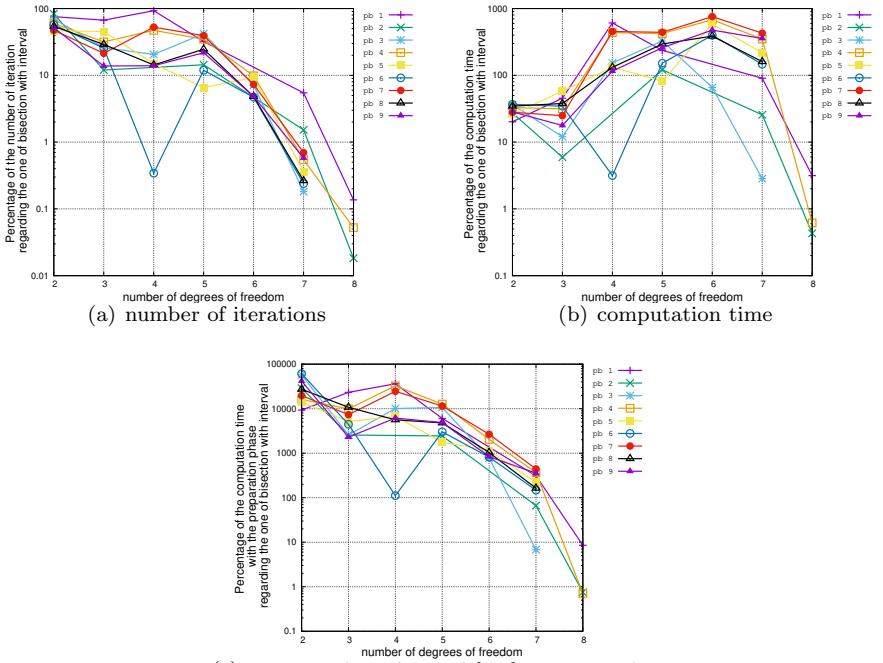
$\{0.4; 1.4\}$ (pb 2,5,8) and $\{0.6; 1.1\}$ (pb 3,6,9). If the computation time exceeds one week, we stop the execution of the algorithm and no result are considered.

4.2.1 Solving CSP

Figures 3, 4 and 5 present the number of iterations and the computation time of bisection using BS, CI and CS methods regarding the performance of BI method for the constraint satisfaction problem with a stopping threshold of 0.01. Figure 6 compares the results of the CS method regarding the ones of BS method. The resolution of the CSP is done as presented in Algorithm 1. Nevertheless, the execution of the algorithm is stopped as soon as one solution is found.

Figure 3(a) proves that the number of iteration required to solve the CSP using BS is lower than the number of iterations using BI. We can deduce that the evaluation of the constraints induces less pessimism using the Bsplines method. Figure 3(c) shows that the BS method (taking into account the preparation phase) is faster than the BI method, but only for a large number of degrees of freedom ($n >= 7$). We highlight that the BS method produces a result for the problems 1 and 2 with 9 degrees-of-freedom in nearly 3 hours whereas the BI method cannot find a solution within one week.

Figure 4 compares the state-of-the-art CI with BI methods and proves that the number of iterations is quite reduced, but with a very larger computation time.



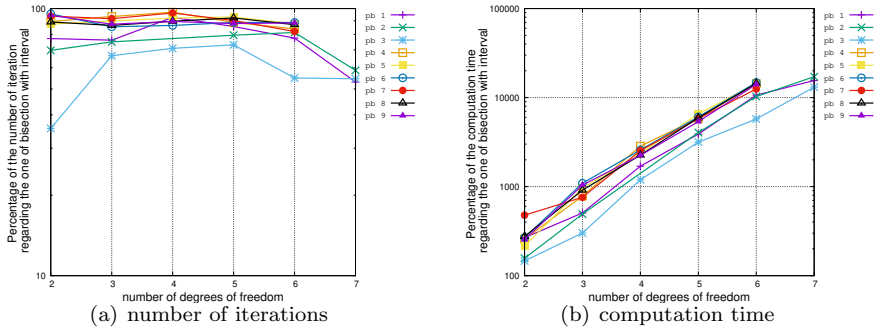
(a) number of iterations

(b) computation time

(c) computation time with the preparation phase

Figure 3: Comparison of the BSplines Bisection and the Interval Bisection methods to solve CSP.

Looking at Figure 5, it seems that the CS methods have nearly the same performances of the BS methods regarding the ones of the BI methods. For an easier comparison, we presents the results of CS method regarding the ones of the BS method in Figure 6. Despite, we believed that the contraction step reduces the boxes before the bisection process, hence reduces the number of iterations of the algorithm, it appears that, for some problems, the number of iterations (and so the computation time) of the CS method is larger than the number of iterations for the BS method. This phenomena is due to the bisection process. During the BS method the diameter of the interval is equal to the initial size of the interval divided by 2^k, with k the number of bisection of this interval. The bisection process choose to bisect the input with the maximal diameter and in case of equality bisect the first input of the queue is bisected. Since all the inputs have the same initial size, this case of equality appears at each iteration. During the CS method, the contraction step may reduce one or several input intervals, hence the case of equality may appear at the beginning of the process and becomes very rare after some iterations. Because of those properties of the bisection process and of the contraction, the BS method and the CS method do not explore the search space in the same order, hence a non-intuitive difference on results could be noticed in terms of computation time and number of iterations.



(a) number of iterations (b) computation time

Figure 4: Comparison of the Interval Contraction and the Interval Bisection methods to solve CSP.

4.2.2 Solving Optimization problem

We also assess our methods on additional optimization problems. We consider the same constraints than in CSP and consider two different cost functions. The cost function of Problem 1 to 9 is the sum of the square joint position $\sum_{i=1}^{n} q_i^2$. However, Problem 11 to 19 consider the sum of square joint torque $\sum_{i=1}^{n} \Gamma_i^2$ as cost function. As for the CSP, the CI method is not suitable for optimization problem. Hence, we choose not to present the results of the CI here for the sake of clarity.

Figure 7 and Figure 8 present the results of the BS method and of the CS method for eighteen complex optimization process regarding the result of the state-of-the-art BS method. As for solving a CSP, the two proposed methods produce a fewer number of iterations, but with a comparable computation time. Figure 9 compares the results of BS and CS methods. Since there is no contraction for the evaluation of the cost function, the two methods seem to present nearly the same performance regarding the number of iterations and the computation time.

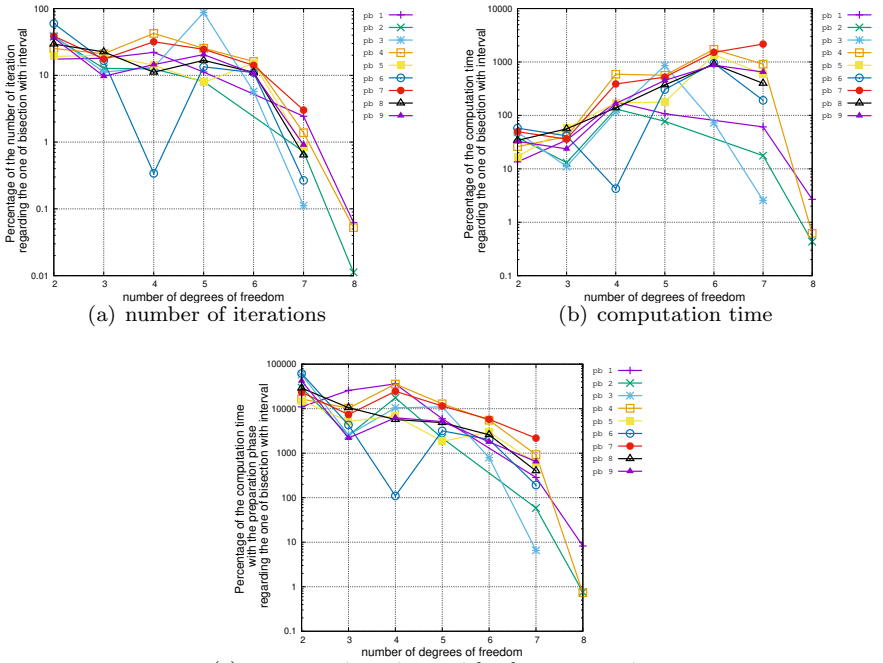
4.3 3D Robot

To assess the effectiveness of our method on more complex systems, we consider the following problem:

$$\begin{array}{rc} \text{Find} & q \in \mathbb{Q} \subset \mathbb{R}^n \\ \text{such as} & \min_q \sum_i q_i^2 \\ \text{with} & x = x_d + [-0.01; 0.01] \\ & y = y_d + [-0.01; 0.01] \\ & z = z_d + [-0.01; 0.01] \\ \forall j = \{1, \ldots, n\} & \Gamma_j \in [-\overline{\Gamma}_j : \overline{\Gamma}_j] \end{array} \tag{43}$$

We present the results for two different 3D robots (with $n = 4$ and $n = 6$ degrees of freedom) which derive from the KUKA LWR as presented on Figure 10. The 4-dof robot is obtained by considering constant the joint number 3 and 5.

The solution of Problem 43 is the static posture, i.e. the joint position, that minimizes the sum of joint angles square and ensures that the end effector position $\{x, y, z\}$ is within a 0.02 meter width box center on a desired position $\{x_d, y_d, z_d\}$ and



(a) number of iterations

(b) computation time

(c) computation time with the preparation phase

Figure 5: Comparison of the BSplines Contraction and the Interval Bisection methods to solve CSP.

that the joint torque Γ_i are within the joint limits. Theses function can be obtained by using the forward kinematic model and the inverse dynamics model as presented in [33].

4.3.1 Constraint Evaluation

Tables 3 and 4 present the performance of the evaluation of the kinematic model for the 4 and 6-dof robots. In order to evaluate the improvement of the BSplines based inclusion function on the constraint evaluation we compute the end-effector box position for several box joint position. One can notice that for the largest ($[-0.5; 1.5]$) interval of the 4-dof robot the natural inclusion function produces less pessimism: this is mainly due to the polynomial approximation of the non-linear functions that produce large error value. It is important to notice that, for nearly all the boxes of the joint position, the BSplines based evaluation produce less pessimism than the natural interval inclusion function.

4.3.2 Solving Optimization problem

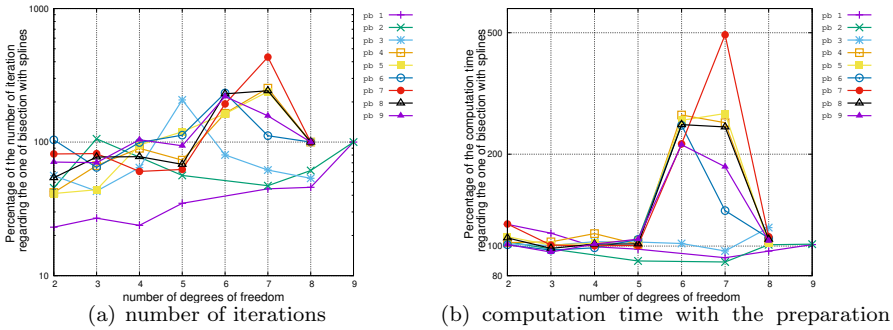
Since the use of the CI method did not provide good results for solving the CSP of planar robots, we do not evaluate this method for 3D robots. In this part, we compare

input	eval	$diam(x)$	$diam(y)$	$diam(z)$
$[-0.5 : 1.5]$	Interval	2.762	2.845	4.968
	BSplines	3.628	3.633	2.070
$[0 : 1]$	Interval	1.554	1.329	2.390
	BSplines	1.523	1.246	0.896
$[0.25 : 0.75]$	Interval	0.903	0.733	1.281
	BSplines	0.099	0.385	0.666
$[0.45 : 0.55]$	Interval	0.191	0.151	0.262
	BSplines	0.106	0.101	0.141
$[0.495 : 0.555]$	Interval	0.019	0.015	0.026
	BSplines	0.010	0.010	0.014

Table 3: Comparison of the evaluation of the end effector $\{X, Y, Z\}$ Cartesian position depending on the diameter of joint position interval for the 4 dof Robot

input	eval	$diam(x)$	$diam(y)$	$diam(z)$
$[-0.5 : 1.5]$	Interval	9.89	10.14	10.19
	BSplines	8.68	8.68	3.86
$[0 : 1]$	Interval	3.389	3.292	3.805
	BSplines	1.053	0.942	1.068
$[0.25 : 0.75]$	Interval	1.503	1.490	1.789
	BSplines	0.146	0.294	0.679
$[0.45 : 0.55]$	Interval	0.283	0.285	0.346
	BSplines	0.137	0.128	0.143
$[0.495 : 0.555]$	Interval	0.028	0.028	0.035
	BSplines	0.015	0.013	0.014

Table 4: Comparison of the evaluation of the end effector $\{X, Y, Z\}$ Cartesian position depending on the diameter of joint position interval for the 6-dof Robot



(a) number of iterations

(b) computation time with the preparation phase

Figure 6: Comparison of the Bsplines Contraction and the Bsplines Bisection methods to solve CSP.

the performance of the BI, BS and CS methods on four different problems for the 4 and 6 dof-robots. The problems 1 and 3 consider a desired end effector position at $\{x, y, z\}$ equal to $\{0.6, 0.6, 0.6\}$ and $\{0.4, 0.4, 0.7\}$ for problems 2 and 4. The problem 1 and 2 consider the nominal maximal torque value and the problem 3 and 4 only 10% of the maximal torque. Consequently, the problem 4 is infeasible, and it is used to assess the performance of our method on infeasible problems.

Tables 5 and 6 emphasize that our methods (BS an CS) solve the optimization problem with less iterations than the state-of-the-art method (BI). They emphasize that the use of the Bsplines properties reduces the number of iterations, due to a less pessimistic evaluation of the constraints. It is also clear that the use of a contraction process (CS) decreases more the number of iterations than pure bisection processes (BI and BS) since it makes possible to reduce the size of the box during the bisection process.

Regarding the computation time, it appears that our method is faster than the state-of-the-art BI method in most of the cases. For the most complex problem (6-dof robot), our method is still the faster one even when we consider the preparation phase (except for one case). It seems that our method is more relevant to prove that the problem has no solution (problem 4) since the number of iterations and the computation time are drastically reduced.

5 Conclusion and perspectives

Interval Analysis is an interesting method that ensures the optimality and the lack of solutions for optimization problems. Unfortunately, it suffers from the pessimism that prohibits its use in complex problems. We propose the use of the BSplines properties and of the Kronecker product in order to reduce the pessimism. We prove that our contribution deals with more complex optimization problems and decreases the computation time. From the given results, our method seems to be the more effective to prove that there is no solution to complex optimization problems.

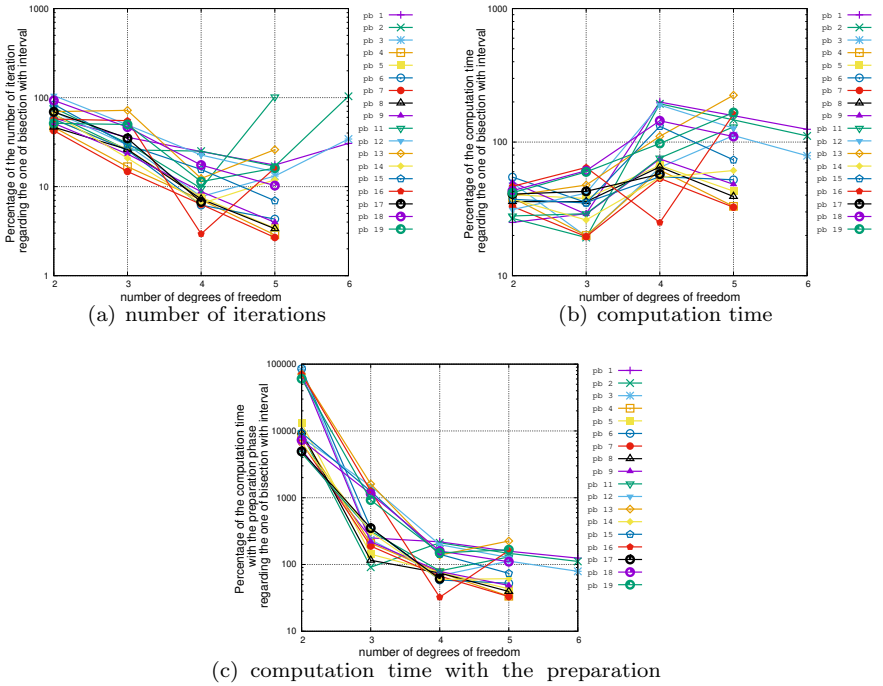
In this paper, we introduced a new formulation of the optimization problem. However, the computation time could be decreased by some improvements. For instance,

problem	solver	number of iterations	computation time (s)	preparation phase (s)	cost function
1	BI	155715	9.51		2.71
	BS	37437	5.42	30.2	2.68
	CS	32331	5.00	27.4	2.68
2	BI	159951	9.64		2.71
	BS	37057	5.36	28.6	2.68
	CS	32015	4.93	23.8	2.68
3	BI	940343	59.5		2.59
	BS	61001	8.17	20.3	1.74
	CS	53733	8.20	29.8	1.74
4	BI	22761	1.39		unfeasible
	BS	1581	0.25	28.6	unfeasible
	CS	1609	0.29	27.8	unfeasible

Table 5: Comparison of the performances of the three solvers for the posture optimization of the 4-dof robot.

problem	solver	number of iterations	computation time (s)	preparation phase (s)	cost function
1	BI	944808895	5-02:12:29		2.43
	BS	116028803	3-08:50:05	455	2.39
	CS	100293163	1-15:07:25	356	2.39
2	BI	1430891169	1-10:09:15		2.43
	BS	111337437	3-11:29:08	715	2.39
	CS	93289815	1-21:36:39	430	2.39
3	BI	72155242357	67-13:54:41		2.51366
	BS	311645729	6-06:41:52	490	1.66
	CS	258134817	3-19:48:31	367	1.66
4	BI	513515683	0-12:25:01		unfeasible
	BS	741335	0-00:27:47	489	unfeasible
	CS	570171	0-00:28:41	368	unfeasible

Table 6: Comparison of the performances of the three solver for the posture optimization of the 6-dof robot.



(a) number of iterations

(b) computation time

(c) computation time with the preparation phase

Figure 7: Comparison of the BSplines Bisection and the Interval Bisection methods to solve optimization problems.

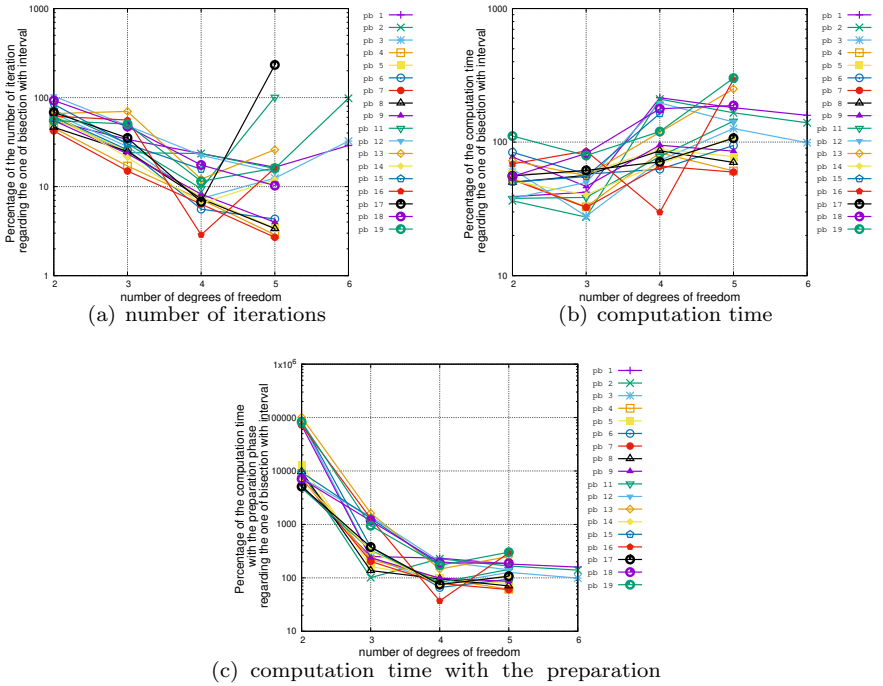
in our work, a very simple bisection process is considered. That is not relevant in some cases. The main avenue of future works will be the use of the control points informations in order to guide the bisection process in a more effective way. Obviously, the use of multi-threading will also be investigated to decrease the computation time. Future works may study the advantages of the BSplines inclusion function to more elaborate optimization algorithms. In addition, we plan to evaluate our method on more complex optimization problems such as robotic motion optimization and to take into account some uncertainties on the models we use.

Acknowledgment

Results were obtained thanks to the support of storage facilities and computing resources of Mésocentre Clermont Auvergne. This work is partially funded by the French Government through the FUI Program (20th call with the project AEROSTRIP).

References

[1] J. Jovic, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, and G. Venture, "Humanoid and human inertia parameter identification using hierarchical opti-



(a) number of iterations

(b) computation time

(c) computation time with the preparation phase

Figure 8: Comparison of the BSplines Contraction and the Interval Bisection methods to solve CSP.

mization," IEEE Transactions on Robotics, vol. 32, no. 3, pp. 726–735, June 2016.

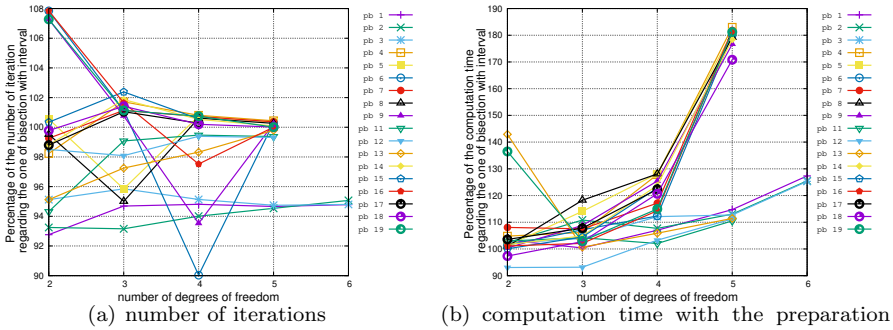
[2] S. Lengagne, J. Vaillant, A. Kheddar, and E. Yoshida, "Generation of whole-body optimal dynamic multi-contact motions," International Journal of Robotics Research, p. 17, Apr 2013, accepted.

[3] J. Nocedal and S. J. Wright, Numerical Optimization. Springer, Aug. 2000.

[4] C. Lawrence, J. L. Zhou, and A. L. Tits, User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints, Electrical Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.

[5] A. Wächter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," Mathematical Programming, vol. 106, pp. 22–57, 2006.

[6] E. Hansen and G. Walster, Global optimization using interval analysis, 2nd ed., M. Dekker, Ed. Marcel Dekker, 2004.



(a) number of iterations

(b) computation time with the preparation phase

Figure 9: Comparison of the BSplines Contraction and the BSplines Bisection methods to solve optimization problems.

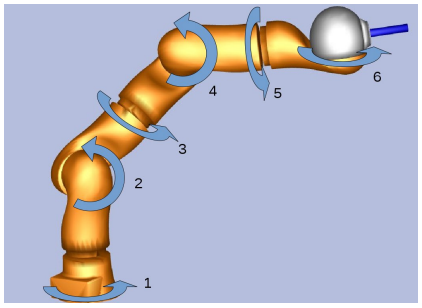


Figure 10: The 6 degrees of freedom robot we use : KUKA LWR

[7] A. Neumaier, "Taylor forms - use and limits." Reliable Computing, 2003.

[8] T. Sunaga, "Theory of interval algebra and its application to numerical analysis," RAAG Memoirs, Ggujutsu Bunken Fukuy-kai, vol. 2, pp. 547–564, 1958.

[9] R. E. Moore and F. Bierbaum, Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics) (Siam Studies in Applied Mathematics, 2.). Soc for Industrial & Applied Math, 1979.

[10] A. Neumaier, Interval methods for systems of equations. Cambridge: Cambridge university press, 1990.

[11] C. Pérez-Galván and I. D. L. Bogle, "Global optimisation for dynamic systems using interval analysis," Computers and Chemical Engineering, 2017.

[12] C. Jiang, X. Han, F. Guan, and Y. Li, "An uncertain structural optimization method based on nonlinear interval number programming and interval analysis method," Engineering Structures, vol. 29, no. 11, pp. 3168 – 3177, 2007.

[13] H. Ma, S. Xu, and Y. Liang, "Global optimization of fuel consumption in j2

rendezvous using interval analysis," Advances in Space Research, vol. 59, no. 6, pp. 1577 – 1598, 2017.

[14] J.-P. Merlet, Interval Analysis and Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 147–156.

[15] L. Jaulin, "Interval analysis and robotics," in SCAN'12, Novosibirsk, Russia, 2012.

[16] S. Rohou, L. Jaulin, L. Mihaylova, F. Le Bars, and S. M. Veres, "Guaranteed computation of robot trajectories," Robotics and Autonomous Systems, vol. 93, pp. 76–84, 2017.

[17] B. Desrochers and L. Jaulin, "Minkowski operations of sets with application to robot localization," SNR'2017, Uppsala., 2017.

[18] D. Benoît and J. Luc, "Computing a guaranteed approximation of the zone explored by a robot," IEEE Transactions on Automatic Control, vol. 62, no. 1, pp. 425–430, 2017.

[19] G. Chabert and L. Jaulin, "Computing the pessimism of inclusion functions," Reliable Computing, vol. 13, no. 6, pp. 489–504, Dec 2007.

[20] L. Netz, "Using horner schemes to improve the efficiency and precision of interval constraint propagation," 2015.

[21] J. Wu, "Uncertainty analysis and optimization by using the orthogonal polynomials," Ph.D. dissertation, 2015.

[22] J. Garloff, C. Jansson, and A. P. Smith, "Lower bound functions for polynomials," Journal of Computational and Applied Mathematics, vol. 157, no. 1, pp. 207 – 225, 2003.

[23] K. Makino and M. Berz, "Taylor models and other validated functional inclusion methods," Int. J. Pure Appl. Math, p. 2003.

[24] N. Revol and F. Rouillier, "Motivations for an arbitrary precision interval arithmetic and the mpfi library," Reliable Computing, vol. 11, no. 4, pp. 275–290, Aug 2005.

[25] L. Jaulin, I. Braems, M. Kieffer, and E. Walter, "Interval methods for nonlinear identification and robust control," in Proceedings of the 41st IEEE Conference on Decision and Control, 2002., vol. 4, Dec 2002, pp. 4676–4681 vol.4.

[26] C. De Boor, A Pratical Guide to Splines. New York: Springer-Verlag, 1978, vol. 27.

[27] R. Kalawoun, S. Lengagne, F. Bouchon, and Y. Mezouar, "Bsplines properties with interval analysis for constraint satisfaction problem: Application in robotics." in 15th International Conference on Intelligent Autonomous Systems IAS-15, june 2018.

[28] K. Schcke, "On the kronecker product," 2013.

[29] C. F. Loan, "The ubiquitous kronecker product," Journal of Computational and Applied Mathematics, vol. 123, no. 1, pp. 85 – 100, 2000, numerical Analysis 2000. Vol. III: Linear Algebra.

[30] R. Kalawoun, "Motion planning of multi-robot system for airplane stripping," Theses, Université Clermont Auvergne, Apr. 2019. [Online]. Available: https://tel.archives-ouvertes.fr/tel-02284448

[31] ALIAS-C++ A C++ Algorithms Library of Interval Analysis for equation Systems. [Online]. Available: https://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++.html

[32] Ibex. [Online]. Available: http://www.ibex-lib.org/doc/intro.html

[33] B. Siciliano and O. Khatib, Springer Handbook of Robotics. Berlin, Heidelberg: Springer-Verlag, 2007.