

BSplines properties with Interval Analysis for Constraint Satisfaction Problem: Application in robotics.

Rawan Kalawoun, Sebastien Lengagne, François Bouchon, Youcef Mezouar

▶ **To cite this version:**

Rawan Kalawoun, Sebastien Lengagne, François Bouchon, Youcef Mezouar. BSplines properties with Interval Analysis for Constraint Satisfaction Problem: Application in robotics.. 15th International Conference on Intelligent Autonomous Systems IAS-15, Jun 2018, Baden-Baden, Germany. pp.490-453, 10.1007/978-3-030-01370-7_39 . hal-01913931

HAL Id: hal-01913931
https://uca.hal.science/hal-01913931

Submitted on 12 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

BSplines properties with Interval Analysis for Constraint Satisfaction Problem: Application in robotics

Rawan Kalawoun[1,2], Sébastien Lengagne[1,3], François Bouchon[1,4], and Youcef Mezouar[1,2]

[1] CNRS, UMR 6602, Pascal Institute, 63171 Aubiére, France
[2] SIGMA Clermont, Pascal Institute, BP10448, 63000 Clermont-Ferrand
[3] Clermont-Auvergne University, Pascal Institute, BP10448, 63000 Clermont-Ferrand, France
[4] Clermont-Auvergne University, Mathematical Laboratory, BP10448, 63000 Clermont-Ferrand

Abstract. Interval Analysis is a mathematical tool that could be used to solve Constraint Satisfaction Problem. It guarantees solutions, and deals with uncertainties. However, Interval Analysis suffers from an overestimation of the solutions, i.e. the pessimism. In this paper, we initiate a new method to reduce the pessimism based on the convex hull properties of BSplines and the Kronecker product. To assess our method, we compute the feasible workspace of a 2D manipulator taking into account joint limits, stability and reachability constraints: a classical Constraint Satisfaction Problem in robotics.

Keywords: Interval Analysis, pessimism, BSplines, Kronecker product, Constraint Satisfaction problem, robot, feasible workspace

1 Introduction

Solving Constraint Satisfaction Problem (CSP) comes down to compute the feasible space of a set of variables that ensures a set of constraints. CSP has many applications in robotics: they are used in planning [1], sequential manipulation planning [2], robot control [3] and are applied to several tasks such as grasping, painting, stripping, etc. In robotics, those constraints may be the robot kinematic and dynamic limits, or the ones imposed by the desired task. Interval Analysis (IA) is a powerful mathematical tool that solve CSP while dealing with uncertainties. In motion generation, it has been shown that finding a new posture in the feasible space is faster than generating it explicitly: IA is an efficient tool to find the feasible space [4]. In parameters estimations, IA ensures the consistency of the results while dealing with some uncertainties [5].

The main drawback of IA is the pessimism, i.e. an overestimation of the solutions. BSplines properties were already used to manage the pessimism in one dimensional case [6]. This paper proposes a new inclusion function for multi-dimensional systems that decreases the pessimism and the computation time using the convex hull property of BSplines and Kronecker product. We assess our method on a reaching task of 2D robot systems with 2, 3, 4 and 5 degrees of freedom while ensuring the system balance.

CSP is presented in Section 2. Section 3 introduces IA, its application in CSP, and its drawback, i.e. the pessimism. Section 4 details the proposed approach to reduce pessimism based on BSplines properties and the Kronecker product. Section 5 details

the technical implementation of our method. The performance of the proposed method is evaluated using a 2D robot with 2, 3, 4, and 5 degrees of freedom in Section 6.

2 Problem Statement

CSP is a mathematical problem searching for a set of states or objects satisfying a certain number of constraints or criteria. CSP could be defined as:

$$\begin{array}{c} \text{Find all} \quad q \in Q \\ \text{such as } \forall j \in \{1,\ldots,m\}\ g_j(q) \leq 0 \end{array} \tag{1}$$

Where:

- $q = \{q_1, ..., q_n\}$: the set of n variables.
- $Q = \{[\underline{q}_1 : \overline{q}_1], ..., [\underline{q}_n : \overline{q}_n]\} \subset \mathbb{R}^n$: the set of variable domains defined by the minimum and maximum values of each variable.
- $g_j(q)$: the j-th constraint equation.

In robotics, q could be the set of joint angles, or the joint trajectory parameters [7], or even the feasible space with bounding errors [4], etc. Moreover, $g_j(q)$ could be non-linear and a piecewise-defined function. It may refer to joint limits, the collision avoidance constraint, the balance constraint, the reachability of the end-effector or manipulability criteria to avoid singularities.

To solve this CSP, four major algorithms can be used: backtracking, iterative improvement, consistency, and IA [8, 9]. Generally, the backtracking algorithm is implemented using its recursive formulation. It can be related to a path in depth of a tree with a constraint on the nodes: as soon as the condition is no longer filled on the current node, the descent on this node is stopped. Iterative improvement algorithms start with a random configuration of the problems and modify it to find the solution. It does not assign an empty solution as an input to the algorithm. However, the consistency in CSP is used to reduce the problem complexity by reducing the search space: it changes the problem formulation in such a way the solutions remain the same. IA is used in CSP to deal with uncertainties, it guarantees finding solutions and avoids the lack of them.

In robotics, CSP is facing inherent uncertainties in the mechanical structure of the robot. Hence, IA is chosen to solve CSP since it deals with uncertainties and it ensures the reliability [7].

3 Interval Analysis

3.1 Presentation

IA was initially developed to take into account the quantification errors introduced by the floating point representation of real numbers with computers [10–12]. IA methods have been also 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 [13–15]. Nowadays, IA is largely used in

robotics [16, 17]. Recent works use IA to compute robot trajectories and the guaranteed explored zone by a robot [18–20].

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 $\mathbb{IR}$. Real arithmetic operations are extended to intervals. Consider an operator $\circ \in \{+, -, *, \div\}$ and $[a]$ and $[b]$ two intervals. Then:

$$[a] \circ [b] = [inf_{u\in[a],v\in[b]}\, u \circ v, \quad sup_{u\in[a],v\in[b]}\, u \circ v] \tag{2}$$

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

$$\mathbf{m}([\mathbf{a}]) = \{\mathbf{m}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{3}$$

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

$$\forall [\mathbf{a}] \in \mathbb{IR}^n,\ \mathbf{m}([\mathbf{a}]) \subseteq [\mathbf{m}]([\mathbf{a}]) \tag{4}$$

An 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. The resulting function is called the natural inclusion function.

3.2 Solving CSP using Interval Analysis

IA can be used to solve CSP as defined in Equation 1. Given input bounds, the algorithm produces a subset of the input space that satisfies all the constraints. This subset is defined as a set of small boxes. A combination between bisection and contraction algorithms solves a CSP using IA. Those two operations are presented hereafter.

Bisection Bisection is an iterative process that decomposes the set of inputs into smaller sets. It can be described as:

1. starting from the initial set $q = Q$,
2. the set of constraints $g(q)$ is computed,
3. q is considered as a feasible box if: $\forall j,\ Sup(g_j(q)) \leq 0$,
4. q is an infeasible box if: $\exists j$ such as $Inf(g_j(q)) \geq 0$,
5. in the other case, i.e. $\exists j$ such as $0 \in g_j(q)$, the current set q is considered as a maybe feasible box and it is split into several sub-boxes q_k. Step 2 is processed considering each q_k.

Those operations are repeated until the size of the sub-boxes is inferior to a defined threshold.

Contraction Contraction is based on the filtering algorithm concept. It manipulates the equation in order to propagate the constraints in two ways: from inputs to outputs and from outputs to inputs. Thus, contraction defines a smaller subset of input boxes respecting the different constraints. For instance, given three variables $x \in [-\infty, 5]$, $y \in$

$[-\infty,4]$ and $z \in [6,\infty]$, and the constraint $z = x+y$, find the intervals of x, y, z respecting this constraint [21]. One can process as follow:

$$\begin{aligned} z = x+y \Rightarrow z \in [z \cap (x+y)] \Rightarrow \quad & z \in [6,\infty] \cap ([-\infty,5]+[-\infty,4]) = [6,9] \\ x = z-y \Rightarrow x \in [x \cap (z-y)] \Rightarrow \quad & x \in [-\infty,5] \cap ([6,9]-[-\infty,4]) = [2,5] \\ y = z-x \Rightarrow y \in [y \cap (z-x)] \Rightarrow \quad & y \in [-\infty,4] \cap ([6,9]-[2,5]) = [1,4] \end{aligned}$$

Thus, by using contractors, the intervals of the variables become tighter: $x \in [2,5]$, $y \in [1,4]$ and $z \in [6,9]$. Contractions generate three types of boxes exactly as bisections: feasible box, infeasible box, and maybe feasible box.

Generally, a combination of contractions and bisections is used to solve CSP. Firstly, contraction reduces the input boxes size. Then, bisection is applied and contraction is called again until the dimension of the generated boxes is smaller than a threshold. This method is the classic contraction/ bisection. It uses the inclusion function **m** during contraction and bisection. Hence, it suffers from pessimism explained in Section 3.3. In Section 4, we propose a new method to decrease pessimism.

3.3 The pessimism

The pessimism overestimates intervals: it produces intervals larger than the real ones. For instance, consider the equation $y = (x+1)^2$ with $x \in [-1,1]$. Using this formulation $y = [0,4]$, but using the following expression $y = x^2 + 2x + 1$, $y = [-1,4]$. Both results are correct, but the solution range may be larger: this phenomena is called pessimism. It is clear that the inclusion function performance depends on the mathematical expression of **m**. Briefly, pessimism is an overestimation of the actual result. It is mainly caused by the multi-occurrence of variables in equations [22, 23]. Each occurrence of the same variable is considered as a different variable relying on the same interval. This article uses BSplines and Kronecker product for the evaluation of the inclusion function in order to reduce pessimism. We must note that decreasing pessimism means finding tighter intervals, though, decreasing the total bisection number. The proposed method to evaluate the inclusion function is shown in Section 4.

4 BSplines identification

BSplines properties were already used to tackle pessimism in one dimension: they are tested on continuous constraints used to optimize robot motion [6]. In this paper, we propose a generalization of this concept to multi-dimensional problems.

4.1 Definition and convex hull property

BSplines function is the weighted sum of several basis functions. It is defined by m control points P_i and basis functions B_i. K is the order of the basis function B_i.

$$F(q) = \sum_{i=1}^{m} B_i^K(q) p_i \tag{5}$$

A BSplines curve is totally inside the convex hull of its control polyline [6]. This property is obtained quite easily from the following definition of a basis function:

$$\forall q \in [\underline{q}, \overline{q}] \;\; \sum_{i=1}^{m} b_i^K(q) = 1 \tag{6}$$

This immediately yields:

$$\forall i \in [1,m] \;\; \underline{F} \leq p_i \leq \overline{F} \Rightarrow \forall q \in [\underline{q}, \overline{q}] \;\; \underline{F} \leq F(q) \leq \overline{F} \tag{7}$$

A conservative estimation of the bounds of $F(q)$ is made based on the minimum and the maximum of the control points. Thus, a bounding box of the considered function can be computed. However, the control points of this function must be identified.

4.2 Multi-dimension BSplines

N-dimensional BSplines are functions defined as:

$$F(q) = \sum_{i=0}^{m_1} \sum_{j=0}^{m_2} ... \sum_{z=0}^{m_n} \left(B_i^{m_1}(q_1) B_j^{m_2}(q_2) ... B_z^{m_n}(q_n) \right) \times P_{i,j,...,z} \tag{8}$$

Where:

- $q = \{q_1, q_2, ..., q_n\} \in [Q] \subset \mathbb{R}^n$,
- m_i is the degree of the input q_i,
- $B^{m_i}(q_i)$ is the BSpline Basis function of degree m_i relied to input q_i,
- $P_{i,j,...,z}$ are the Control Points grouped into vector P.

As in the one-dimensional case, the BSpline curve is entirely in the convex hull of its control polyline:

$$\forall q \in [Q] \;\; F(q) \in [\min(P), \max(P)] \tag{9}$$

4.3 Control Point identification

One dimensional case Considering that all the functions can be described as a polynomial expression of the input as done in [6], we have :

$$\forall q \in [\underline{q}, \overline{q}] \; F(q) \in \sum_{i=0}^{n} a_i \times q^i \tag{10}$$

where $\{a_0, a_1, \ldots, a_n\} \in \mathbb{R}^{n+1}$ are the coefficients of the polynomial. This equation can be written as :

$$F(q) = [1, q, \ldots, q^n] \times [a_0, a_1, \ldots, a_n]^T \tag{11}$$

and knowing the coefficients a_i, the coefficients p_i of the equivalent control point are computed, such as:

$$F(q) = [1, q, \ldots, q^n] \times \mathbf{B} \times [p_0, p_1, \ldots, p_n]^T \tag{12}$$

where $\mathbf{B}$ is a matrix that contains the polynomial parameters of the BSplines basis functions. Therefore, we can compute the corresponding BSplines parameters as:

$$[p_0, p_1, \ldots, p_n]^T = \mathbf{B}^{-1} \times [a_0, a_1, \ldots, a_n]^T \tag{13}$$

N-dimensional case The same procedure can be applied to the N-dimensional case. Let us note P the vector of the control points and X the vector of the polynomial coefficients. P and X are related through the following equation:

$$P = \mathbb{B}^{-1}X \tag{14}$$

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

$$\mathbb{B} = \mathbf{B}_1 \otimes \mathbf{B}_2 \otimes \ldots \otimes \mathbf{B}_i \otimes \ldots \otimes \mathbf{B}_n \tag{15}$$

$\mathbf{B}_i$ are matrices linking the control point to the coefficient of the polynomial expression of the basis functions of input q_i and $\otimes$ is the Kronecker product as defined hereafter.

4.4 Kronecker product

Definition The Kronecker product was firstly studied in the nineteenth century [24]. The Kronecker product of two matrices $A \in \mathbb{R}^{n_1,m_1}$ and $B \in \mathbb{R}^{n_2,m_2}$ is the matrix $A \otimes B \in \mathbb{R}^{n_1 \times n_2, m_1 \times m_2}$ defined as follows:

$$A = \begin{pmatrix} a_{1,1} & \ldots & a_{1,n_1} \\ \vdots & \vdots & \vdots \\ a_{m_1,n_1} & \ldots & a_{m_1,n_1} \end{pmatrix} \quad A \otimes B = \left(\begin{array}{c|c|c|c} a_{1,1} \times B & a_{1,2} \times B & \ldots & a_{1,n_1} \times B \\ \hline a_{2,1} \times B & a_{2,2} \times B & \ldots & a_{2,n_1} \times B \\ \hline \vdots & \vdots & \vdots & \vdots \\ \hline a_{m_1,1} \times B & a_{m_1,2} \times B & \ldots & a_{m_1,n_1} \times B \end{array}\right) \tag{16}$$

4.5 Properties of the Kronecker Product

More properties of the Kronecker product can be found in [25]. Here we focus on the associative (Eq. 17) and the invertible (Eq. 18) properties :

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \tag{17}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{18}$$

Using Equation (18), Equations (14) and (15) can be turned into :

$$P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} \otimes \ldots \otimes \mathbf{B}_n^{-1}\right) X \tag{19}$$

Assuming $\mathbb{B} \in \mathbb{R}^{x,x}$, with $x = \prod_{k=1}^{n} m_i$, the complexity of the inversion of the matrix $\mathbb{B}$ in Equation (14) is $\mathcal{O}(x^3)$. Using the invertible property, the complexity decreases to $\mathcal{O}(x \sum_n m_i^2)$ where $x = \prod_n m_i$, that will allow faster computation. Hence, Equation (19) is used to compute control points.

One can consider the following property relating matrix/vector multiplication and the Kronecker product:

$$A.X = (I \otimes A)X \tag{20}$$

Where A is a matrix and X is a vector. Equation (20) is used to reduce computation time as it will be explained hereafter.

4.6 Recursive Inverse Kronecker Product

Using Equation (20), Equation (19) could be written as following:

$$\begin{aligned} P &= \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1} \otimes \ldots \otimes \mathbf{B}_n^{-1}\right) X_n \\ &= \left(\mathbf{B}_1^{-1} \otimes \ldots \otimes \mathbf{B}_{n-1}^{-1}\right) \left(I \otimes \mathbf{B}_n^{-1}\right) X_n \\ &= \left(\mathbf{B}_1^{-1} \otimes \ldots \otimes \mathbf{B}_{n-1}^{-1}\right) X_{n-1} \\ \text{With: } X_{n-1} &= \left(I \otimes \mathbf{B}_n^{-1}\right) X_n \end{aligned} \tag{21}$$

This equation can be used recursively to reduce the computation time. Regarding the low dimension of the matrix $\mathbf{B}_i$, the inversion is done numerically. However, future works will address this issue as presented in Section 7. As an example, consider two matrices $A \in \mathbb{R}^{2\times 2}$ and $B \in \mathbb{R}^{2\times 2}$, and $X = [X_1, X_2] \in \mathbb{R}^4$ with $\{X_1, X_2\} \subset \mathbb{R}^2$, the following product can be computed such as:

$$(A \otimes B)X = \begin{bmatrix} a_{1,1}.B.X_1 + a_{1,2}.B.X_2 \\ a_{2,1}.B.X_1 + a_{2,2}.B.X_2 \end{bmatrix} \tag{22}$$

Despite the multi-occurrence of $B.X_1$ and $B.X_2$, $B.X_1$ and $B.X_2$ are calculated only once.Though, the computation time is decreased.

4.7 Example

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

$$f(q_1, q_2) = 1 - 3q_1 - 2q_2 + 4q_1q_2$$

with $q_1, q_2 \in [-10, 10]$. Thus, we have $X = [1, -3, -2, 4]^T$ and $\mathbf{B}_1 = \mathbf{B}_2 = \begin{bmatrix} 0.5 & 0.5 \\ -0.05 & 0.05 \end{bmatrix}$.
We can compute the equivalent control point using:

$$P = \left(\mathbf{B}_1^{-1} \otimes \mathbf{B}_2^{-1}\right) X = [451, -389, -409, 351]^T \tag{23}$$

Thus using our method, we can deduce that $f \in [-409, 451]$ whereas using the natural inclusion functions we obtain $f \in [-449, 451]$. Using a 3D plot of f, we can deduce that f is limited between -409 and 451. Once can deduce that our method reduces pessimism in this example. 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.

5 BSplines and Kronecker product in the contraction:

In this section, we present how to use the convex hull properties of BSplines and the Recursive Inverse Kronecker Product in the contraction process. Starting from a given box $[q]$, the algorithm will perform the contraction on each constraint $g_j(q)$ while it does not produce an empty set on the constraint or on one input. For each contraction, the minimum and the maximum of each variable q_i are evaluated, and tighter q_i intervals are found.

5.1 One constraint contraction

Let us consider a constraint $g_j(q) \leq 0$, our goal is to find the set of q intervals respecting this constraint. Let $g_j(q)$ be the expression of the constraint j and υ_j be a new variable initialized to $[-\infty, 0]$ (since $g_j(q) \leq 0$). We create an extended equation of this constraint $f_j(q, \upsilon_j)$ such as:

$$f_j(q, \upsilon_j) = g_j(q) - \upsilon_j \tag{24}$$

The contraction will be applied on the new polynomial $f_j(q, \upsilon_j)$. This polynomial is a sum of many monomials μ_i. If we consider that the bounds of $g_j(q)$ can be found, using $\mathbb{B}$ and X, through the minimum and maximum value of the vector P (Equation 14), thus the bounds of $f_j(q, \upsilon_j)$ can be found by the minimum and maximum value of P_e such as:

$$P_e = \mathbb{B}_e^{-1} X_e \tag{25}$$

With :

$$\mathbb{B}_e = \mathbf{B}_\upsilon \otimes \mathbb{B} \tag{26}$$

$$X_e = [X^T, -1, 0, \ldots, 0]^T \tag{27}$$

With $\mathbf{B}_\upsilon$ a 2×2 matrix since υ_j is only of order 1 in Equation (24).

Considering x_i the $i-th$ element of vector X_e that corresponds to the coefficient of the monomial μ_i of Equation 24, the bounds of the monomial μ_i can be computed, for all $x_i \neq 0$ using:

$$\mu_i \in [Inf(\rho_i); Sup(\rho_i)] \tag{28}$$

$$\text{with } \rho_i = -\frac{1}{x_i} \left(\mathbb{B}_e^{-1} X_{e,i} \right) \tag{29}$$

With $X_{e,i}$ equals to X_e except for the $i-th$ component that equals to zero.

5.2 Monomial contraction:

Considering a monomial $\mu_i = \prod_{\forall k} \sigma_{i,k}$ composed of the multiplication of several input intervals $[\sigma_{i,k}]$, if no input interval contains zero, the input interval can be contracted thanks to:

$$[\sigma_{i,k}] = [\sigma_{i,k}] \cap \frac{[Inf(\rho_i); Sup(\rho_i)]}{\prod_{\forall k \neq i} \sigma_{i,k}} \tag{30}$$

The current input set will be considered as unfeasible if this intersection returns an empty interval.

5.3 Implementation trick

The Recursive Inverse Kronecker Product requires a non negligible computation time, and hence must be used in a smart way. Hence, instead of Equation 29, we rather implement the following Equation:

$$\rho_i = -\frac{1}{x_i} \left(\mathbb{B}_e^{-1} X_e + \mathbb{B}_e^{-1} X_i \right) \tag{31}$$

Considering $X_i + X_e = X_{e,i}$, the first part of the Equation ($\mathbb{B}_e^{-1} X_e$) is computed once for all the monomials of the considered constraint, whereas the structure of X_i (only one non-zero value) makes possible faster computation of $\mathbb{B}_e^{-1} X_i$.

5.4 How to deal with non linear functions ?

The proposed method assumes that the constraints can be formulated as polynomial equations of the inputs. Nevertheless, robotics equations are generally non-linear (using sine and cosine). Here we proposed to create intermediate inputs for $\cos(q_i)$ and $\sin(q_i)$. Each time those intermediate inputs are contracted, they propagate the modification to the input q_i that re-propagate on $\cos(q_i)$ and $\sin(q_i)$.

6 Simulations and results

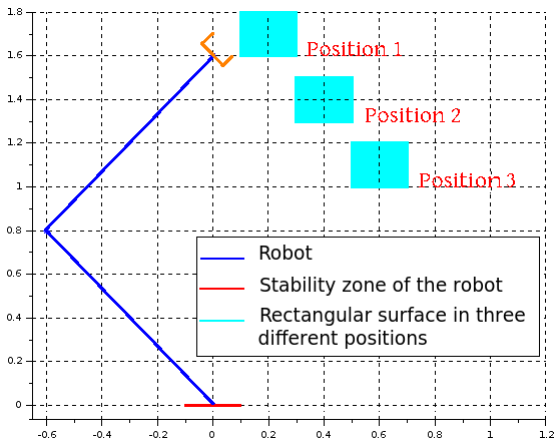


Fig. 1: Two dof Robot with a square surface in three different positions and a robot stability margin

We tested our BSplines IA algorithm on 2D-robots with n degrees of freedom. The feasible space of the joint values is computed using our method. Hence, a set of joints boxes, i.e. intervals, of the robot respecting a set of constraints is defined. In this application, the end-effector must be inside a square surface while the quasi-static balance of the robot is guaranteed though the projection of the center of mass. We show the results of the implementation of the proposed method for three different positions of the square surface as presented in Figure 1, with a robot of 2, 3, 4 and 5 degrees of freedom for each position. We consider the 2D robot with a total length of 2, with all the segment of equal values $(2/n)$ and equal mass, a root position in (0,0) and with all joint limits of $[-1.5, 1.5]$. The size of the desired square surface is 0.2 with two feasible positions $(0.2; 1.7)$, $(0.4; 1.4)$ and one unfeasible position $(0.6; 1.1)$ (due to balance constraint).

The balance of the robot is considered by ensuring that the projection of its center of mass remains in the interval $[-0.1, 0.1]$ (considering that the center of mass of each segment is at the middle of the segment).

The surface reachability decision is based on the position of the end-effector. This position is computed using a composition of transformation matrices. The transformation matrix of q_i is defined by: $T_{q_i} = \begin{pmatrix} cos(q_i) & -sin(q_i) & \beta_i \\ sin(q_i) & cos(q_i) & 0 \\ 0 & 0 & 1 \end{pmatrix}$ Where $\forall i > 1, \beta_i = \frac{l}{n}$ and $\beta_0 = 0$, since the initial robot position is along y axis. Hence the position of the end-effector P_f= (T_f(0,2), T_f(1,2)) Where $T_f = T_{\pi/2} T_{q_1} ... T_{q_i} ... T_{q_n}$.

We assign 0.01 to the box threshold during the bisection process. We compare three methods: the classic contraction with the bisection (Solver 1), the BSplines contraction of the monomial that are composed only of input variables with the bisection (Solver 2), and the BSplines contraction of all the monomials in the constraints equations with the bisection (Solver 3).

The number of iterations and the computation time required to find the feasible workspace are shown in Figures 3a and 3b respectively, for a threshold of 0.01 and in Figures 2a and 2b respectively, for a threshold of 0.1. The computing time required to find the feasible workspace for a robot of 5 dof using a precision of 0.01 was very huge: more than one week. Hence, we show the results of a robot with 1, 2, 3, and 4 dof for a precision of 0.01.

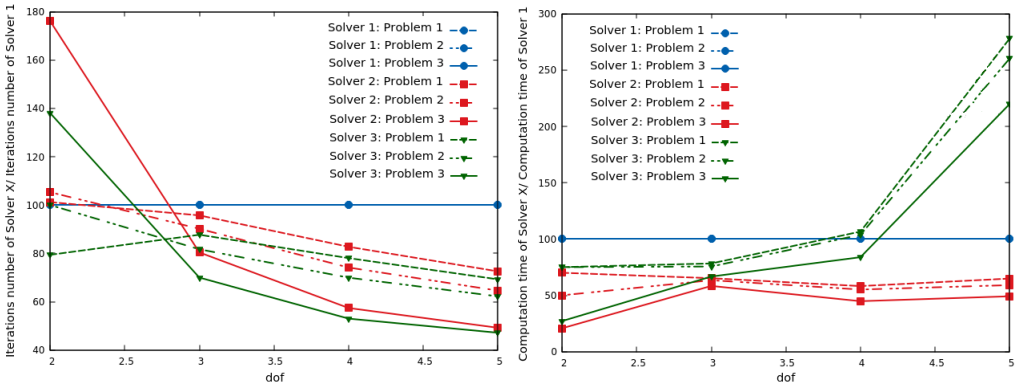
For both threshold, the number of iterations using the BSplines contraction with the bisection (Solver 2, 3) is smaller than the number of iterations in the state-of-the-art method (Solver 1). Though, BSplines IA uses less number of bisections to find the robot feasible workspace. Hence, our method decreases pessimism.

A comparison between the two solvers using the BSplines contraction (Solver 2 and 3) shows that (Solver 3) has lower number of iterations than (Solver 2). Though, the contraction of each monomials of the constraint equation helps to reduce pessimism. However, (Solver 3) is slower than (Solver 2), since doing better contractions requires more computation time (see Figure 3b, 2b). It is clear that while increasing the accuracy (decreasing the threshold), our method (Solver 2) becomes faster than the classic method (Solver 1). Hence the BSplines contraction/ bisection reduces pessimism, and it is faster than the contraction/ bisection for high accuracy. We proposed two solvers using BSplines contraction: Solver 2 is faster than Solver 3 and 1, but Solver 3 reduces the pessimism more than Solver 2.

Tables 1 and 2 shows the number of iterations and the computing time required to find the feasible space of our different problems for a precision of 0.01.

7 Conclusion and perspectives

In robotics, Constraint Satisfaction Problem is used to compute the feasible space of the set of robot joint angles: this computation is still an open research domain. In this paper, Interval Analysis is used to solve Constraint Satisfaction Problem based on bisection and contraction concept. This technique suffers from pessimism. Hence, we propose a new way to represent the inclusion function that decreases pessimism. Our



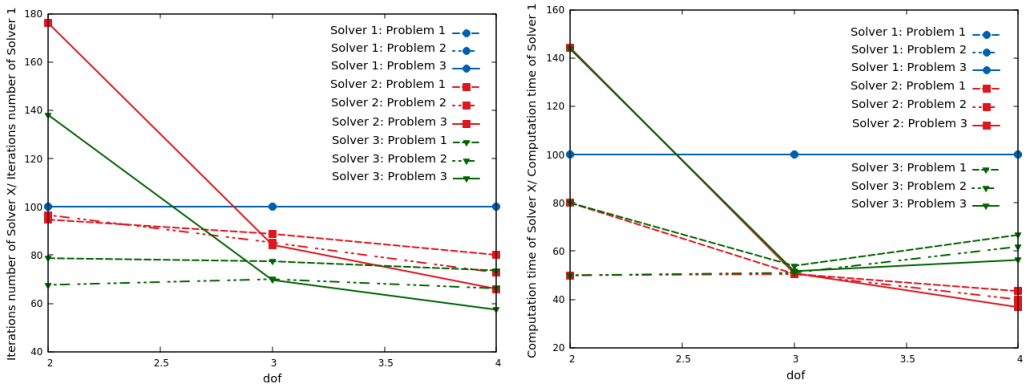
(a) Iteration numbers (b) Computation time

Fig. 2: Iteration numbers and computation time of the three different solvers compared to Solver 1 for a precision of 0.1

method is based on the convex hull properties of BSplines functions and the resolution of an iterative Kronecker product. We assessed our method on different 2D cases and emphasize that it decreases the number of iterations and it may decrease the computation time. In future works, we do believe that the computation time can be reduced by solving the Kronecker inverse recursive product without inverting all the matrices. In the current implementation used in this paper, the bisection process is performed by splitting the largest input interval into two equal intervals. We plan to implement more efficient bisection process by bisecting the input interval that has the most significant impact on unsatisfied constraint. Eventually, the proposed method can be applied to 3D robot taking into account kinematic constraints such as collision and self-collision avoidance and also singularity avoidance or more dynamic constraints such balance or torque limits.

Position	Solver \ DOF	2	3	4
1	1	1975	2288339	1356748167
	2	1871	2032347	1086632883
	3	1555	1771783	999374305
2	1	705	466451	737351907
	2	681	397469	536515897
	3	477	326615	488179609
3	1	21	74895	231273567
	2	37	63129	152641419
	3	29	52201	132694819

Table 1: The number of incrementation required to solve Position 1 problem using Solver 1, Solver 2 and Solver 3/ PRECISION 0.01



(a) Iteration numbers (b) Computation time

Fig. 3: Iteration numbers and computation time of the three different solvers compared to Solver 1 for a precision of 0.01

Position	Solver \ DOF	2	3	4
1	1	0 / 00 : 00 : 0.05	0 / 00 : 03 : 39	4 / 14 : 39 : 57
	2	0 / 00 : 00 : 0.04	0 / 00 : 01 : 51	2 / 00 : 02 : 09
	3	0 / 00 : 00 : 0.04	0 / 00 : 01 : 58	3 / 01 : 50 : 30
2	1	0 / 00 : 00 : 0.02	0 / 00 : 00 : 43	2 / 12 : 11 : 18
	2	0 / 00 : 00 : 0.01	0 / 00 : 00 : 21	1 / 00 : 02 : 35
	3	0 / 00 : 00 : 0.01	0 / 00 : 00 : 22	1 / 13 : 14 : 06
3	1	0 / 00 : 00 : 0.0005	0 / 00 : 00 : 07	0 / 19 : 08 : 45
	2	0 / 00 : 00 : 0.0007	0 / 00 : 00 : 03	0 / 07 : 02 : 04
	3	0 / 00 : 00 : 0.0007	0 / 00 : 00 : 035	0 / 10 : 47 : 06

Table 2: The time (in days / HH:MM:SS) required to solve Position 1 problem using Solver 1, Solver 2 and Solver 3/ PRECISION 0.01

References

1. N. N. W. Tay, A. A. Saputra, J. Botzheim, and N. Kubota, "Service robot planning via solving constraint satisfaction problem," ROBOMECH Journal, vol. 3, no. 1, p. 17, 2016.
2. T. Lozano-Pérez and L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sept 2014, pp. 3684–3691.
3. M. P. Fromherz, T. Hogg, Y. Shang, and W. B. Jackson, "Modular robot control and continuous constraint satisfaction," in Proc. IJCAI-01 Workshop on Modelling and Solving Problems with Constraints, 2001, pp. 47–56.
4. S. Lengagne, N. Ramdani, and P. Fraisse, "Planning and fast replanning safe motions for humanoid robots," IEEE Transactions on Robotics, vol. 27, no. 6, pp. 1095 –1106, dec. 2011.
5. L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics. Springer London Ltd, Aug. 2001.

6. S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of Whole-body Optimal Dynamic Multi-Contact Motions," The International Journal of Robotics Research, p. 17, Apr. 2013.
7. J.-P. Merlet, "Interval analysis and reliability in robotics," International Journal of Reliability and Safety, vol. 3, no. 1-3, pp. 104–130, 2009.
8. M. Yokoo and K. Hirayama, "Algorithms for distributed constraint satisfaction: A review," Autonomous Agents and Multi-Agent Systems, vol. 3, no. 2, pp. 185–207, Jun 2000.
9. E. Gelle and B. Faltings, "Solving mixed and conditional constraint satisfaction problems," Constraints, vol. 8, no. 2, pp. 107–141, 2003.
10. T. Sunaga, "Theory of interval algebra and its application to numerical analysis," RAAG Memoirs, Ggujutsu Bunken Fukuy-kai, vol. 2, pp. 547–564, 1958.
11. 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.
12. A. Neumaier, Interval methods for systems of equations. Cambridge: Cambridge university press, 1990.
13. C. Pérez-Galván and I. D. L. Bogle, "Global optimisation for dynamic systems using interval analysis," Computers and Chemical Engineering, 2017.
14. 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.
15. 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.
16. J. P. Merlet, Interval Analysis and Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 147–156.
17. L. Jaulin, "Interval analysis and robotics," in SCAN'12, Novosibirsk, Russia, 2012.
18. 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.
19. B. Desrochers and L. Jaulin, "Minkowski operations of sets with application to robot localization," SNR'2017, Uppsala., 2017.
20. 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.
21. F. Le Bars, A. Bertholom, J. Sliwka, and L. Jaulin, "Interval SLAM for underwater robots; a new experiment," in NOLCOS 2010, France, Sep. 2010, p. XX.
22. J. Wu, "Uncertainty analysis and optimization by using the orthogonal polynomials," Ph.D. dissertation, 2015.
23. L. Netz, "Using horner schemes to improve the efficiency and precision of interval constraint propagation," 2015.
24. K. Schäcke, "On the kronecker product," 2013.
25. 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.