Exact interval propagation for the efficient solution of position analysis problems on planar linkages

Enric Celaya*, Tom Creemers, and Lluís Ros

Institut de Robòtica i Informàtica Industrial (CSIC-UPC)
Llorens Artigas 4-6, 08028 Barcelona, Spain
E-mails: {celaya,creemers,ros@iri.upc.edu}

Abstract

This paper presents an interval propagation algorithm for variables in planar single-loop linkages. Given intervals of allowed values for all variables, the algorithm provides, for every variable, the whole set of values, without overestimation, for which the linkage can actually be assembled. We show further how this algorithm can be integrated in a branch-and-prune search scheme, in order to solve the position analysis of general planar multi-loop linkages. Experimental results are included, comparing the method's performance with that of previous techniques given for the same task.

Keywords: Interval propagation, planar linkages, box approximation, loop equation, position analysis, forward and inverse kinematics.

1. Introduction

In recent years there has been a growing interest in the use of interval methods in Kinematics, specially for the task of finding the feasible configurations of complex linkages—the so-called position analysis problem—which, when performed analytically, may involve a large number of equations or high-degree polynomials [1–3]. When the solution space is continuous, interval methods can provide a discrete approximation of such space as a set of enclosing boxes, which can, at least in principle, be refined as much as desired [4]. Advantages of interval methods include their completeness, the

*Corresponding author. Phone: +34 93 401 5787

fact that they do not require complex algebraic manipulations, and that the solution set is not populated with imaginary values.

Interval methods for position analysis of linkages usually rely on a branch-and-prune strategy: given a set of intervals for the involved variables, which determine an initial box where solutions are to be sought, a pruning method is applied to the box to eliminate regions of the box that cannot contain a solution. When the box cannot be further reduced, it is split, usually into two halves (or branches), each of which is treated recursively by the same process. The pruning method may be a general interval propagation algorithm applied to the kinematic equations of the linkage [1, 4–7], or a more specific method suited to the problem [2, 3, 8].

Often, interval methods are, by design, subject to overestimation [9]. This fact prevents reducing a box as much as possible before branching, which may result in important efficiency losses. There is a trade-off between the accuracy of the pruning process and its efficiency: an excessive complexity of the pruning algorithm may overshadow its better accuracy in the reduction of a box, resulting in a globally less efficient algorithm.

In this paper we present an exact, though simple, interval propagation algorithm for variables in planar single-loop linkages with revolute joints. The algorithm provides the exact angle intervals for which a solution exists, assuming that the remaining angles can vary within specified ranges. By "exact", we mean here that the intervals are obtained with no over- or under-estimation, at least in the ideal case of performing all computations with infinite precision. Note however that, in practice, actual computations almost always introduce approximations which prevent obtaining mathematically exact results. On this regard, the tools of Interval Analysis [9] provide a means to perform computations with approximate values and still make sure that the result will be enclosed in a certain interval. Our algorithm can indeed be implemented using such tools to provide guaranteed intervals, however, since this is not the focus of the paper, we implemented the algorithm using standard, floating-point computations. In fact, in the context of Interval Analysis, our propagation algorithm constitutes an *optimal contractor* [4] for equations of planar single-loop linkages with interval-constrained variables, providing the smallest box that contains the solution set of such a constraint satisfaction problem.

The paper also shows how the previous propagation algorithm may be embedded within a branch-and-prune process as described above, in order to solve complex, multi-loop linkages. For clarity purposes, the analysis is

limited to planar linkages with revolute joints only, but the method can also be extended to deal with slider joints, and equivalent results can also be obtained for spherical linkages [10].

The paper is organized as follows. Section 2 considers the case of single-loop linkages and derives the exact intervals containing all joint angles for which the loop closes, assuming all joints can freely rotate within the $[0, 2\pi]$ range. The following three sections compute the corresponding intervals assuming that further constraints hold on the joint angles: Section 3 assumes one joint is held fixed, Section 4 assumes one joint can only move within a sub-interval of $[0, 2\pi]$, and Section 5 assumes all joints move within different sub-intervals each. Based on these results, Section 6 presents a branch-and-prune algorithm for the position analysis of multi-loop linkages. Section 7 shows some experiments on an implementation of this algorithm. Finally, Section 8 concludes the paper and highlights points deserving further attention.

2. Feasibility ranges in single-loop linkages

Consider a single-loop planar linkage formed by n links of positive lengths $l_1, \ldots, l_n$, cyclically connected through revolute joints (Fig. 1). It is well

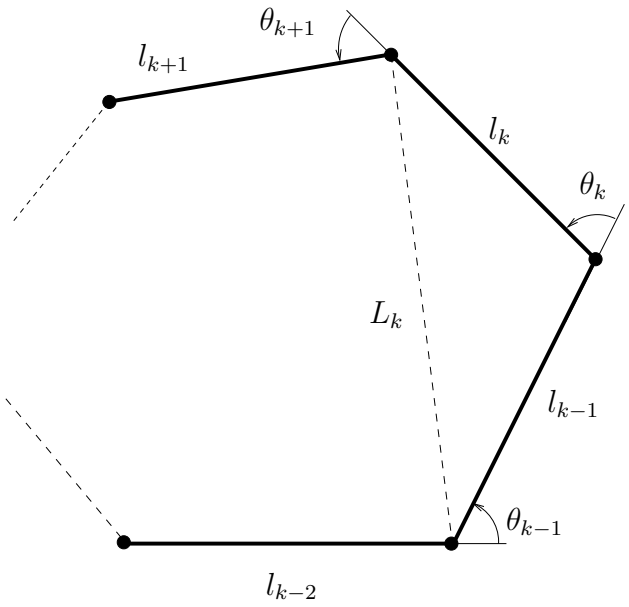


Figure 1: A single-loop planar linkage with revolute joints.

known that the n joint angles $\theta_1, \ldots, \theta_n$ define a feasible configuration of the linkage if, and only if, they satisfy the loop equation

$$\mathbf{R}(\theta_1)\ \mathbf{Tx}(l_1)\ \ldots\ \mathbf{R}(\theta_n)\ \mathbf{Tx}(l_n) = \mathbf{I}, \tag{1}$$

where $\mathbf{R}(\theta)$ and $\mathbf{Tx}(l)$ denote homogeneous matrices encoding a rotation of angle θ, and a translation of distance l along the x axis, respectively.

Our goal is to determine the *feasibility range* for each angle θ_k in Eq. (1), i.e., the set of values for θ_k for which Eq. (1) has a solution, allowing the n links to form a closed polygon. Hereafter, we will use the symbols e_n and $\mathcal{S}_k(e_n)$ to refer to Eq. (1), and to the feasibility range for θ_k in this equation, respectively. All indices will be understood modulus n, so that, e.g., index $n+1$ corresponds to index 1, and index 0 corresponds to index n.

Eq. (1) is solvable, and hence the ranges $\mathcal{S}_k(e_n)$ are non-empty, whenever the lengths $l_1, \ldots, l_n$ verify a certain condition, called the *closure condition* for the links. We next derive this condition for different values of n, and show how the feasibility ranges $\mathcal{S}_k(e_n)$ can be determined in each case.

For $n = 2$, the closure condition is simply $l_1 = l_2$, in which case the feasibility ranges for θ_1 and θ_2 reduce to a single point, i.e.:

$$\mathcal{S}_1(e_2) = \mathcal{S}_2(e_2) = \begin{cases} \{\pi\}, & \text{if } l_1 = l_2, \\ \varnothing, & \text{if } l_1 \neq l_2. \end{cases}$$

For $n = 3$, the closure condition is given by the triangle inequalities

$$|l_1 - l_2| \leq l_3 \leq l_1 + l_2, \tag{2}$$

and the exterior angles of the triangle can be found by using the cosine rule. For the angle θ_2, for example, the rule dictates

$$l_3^2 = l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2), \tag{3}$$

so that

$$\theta_2 = \pm \arccos\left(\frac{l_3^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right). \tag{4}$$

Eq. (4) provides real values only when the argument of the arccosine function is in the interval $[-1, 1]$ or, equivalently, when the triangle inequalities (2) hold. In this case, the feasibility range for θ_2 consists of two points. Denoting by θ_2^- and θ_2^+ the two determinations provided by Eq. (4), we have:

$$\mathcal{S}_2(e_3) = \begin{cases} \{\theta_2^-, \theta_2^+\} & , \text{ if } |l_1 - l_2| \leq l_3 \leq l_1 + l_2, \\ \varnothing & , \text{ otherwise.} \end{cases}$$

The two determinations θ_2^- and θ_2^+ coincide when the argument of the arccosine function is $+1$ or -1, in which case $\mathcal{S}_2(e_3)$ reduces to a single point. The feasibility ranges $\mathcal{S}_1(e_3)$ and $\mathcal{S}_3(e_3)$ can be obtained in an analogous way.

For $n > 3$, Eq. (1) can have an infinite number of solutions in general. In this case, an expression of the closure condition that is convenient for our purposes can be formulated in terms of the length L_k of the segment connecting joints $k-1$ and $k+1$ (Fig. 1). Since the subchain formed by links $k-1$ and k constrains L_k to the interval

$$I_k = [\,|l_{k-1} - l_k|,\, l_{k-1} + l_k], \tag{5}$$

the condition for the linkage to form a closed polygon is that the end-points of the subchain formed by the remaining $n-2$ links $i \notin \{k-1, k\}$ can also reach some length in the interval I_k. To determine when this is possible, we prove the following.

Lemma 1. *The set of lengths reachable by a chain of m links of lengths $l_1, \ldots, l_m$ articulated through revolute joints is the interval*

$$A = [A_{min}, A_{max}], \tag{6}$$

with

$$A_{max} = \sum_{i=1}^{m} l_i, \tag{7}$$

and

$$A_{min} = \max\{0,\, l_M - \sum_{\substack{i=1,\\ i \neq M}}^{m} l_i\}, \tag{8}$$

where l_M is the largest l_i, i.e., $l_M \geq l_i, i = 1, \ldots, m$.

Proof. It is easy to see that for $m = 1$ and $m = 2$ the lemma holds: For $m = 1$ we get $A = [l_1, l_1]$, so that the lemma is trivially true. For $m = 2$ we get $A = [\,|l_1 - l_2|,\, l_1 + l_2]$, which corresponds to the result established in (5). The proof for $m \geq 3$ is by induction: We assume that the lemma is true for $m-1$ links, and we prove it for m links.

Clearly, the maximum length is reached when the chain is fully stretched, which corresponds to the upper bound A_{max}. If we vary one or more of the θ_i, this length will vary continuously, with a minimum value that we will determine next. We distinguish two cases:

a) $l_M > \sum_{i=1,i\neq M}^{m} l_i$. In this case the lower bound A_{min} is $l_M - \sum_{i=1,i\neq M}^{m} l_i$. Such value can be reached by making all $\theta_i = 0$, except for the angles adjacent to link M, for which we make $\theta_{M-1} = \theta_M = \pi$, and is clearly the minimum possible.

b) $l_M \leq \sum_{i=1,i\neq M}^{m} l_i$. In this case, the lower bound A_{min} is 0, and we must prove that it can be always reached by the m-link chain. This amounts to proving that the length l_M can be reached by the chain of the remaining $m-1$ links. By induction hypothesis, the chain with $m-1$ links $i \neq M$, can reach any length in the interval

$$A' = \left[\max\{0,\ l_{M'} - \sum_{\substack{i=1,\\ i\neq M, M'}}^{m} l_i\},\ \sum_{\substack{i=1,\\ i\neq M}}^{m} l_i \right], \tag{9}$$

where $l_{M'}$ is the largest of the $m-1$ link lengths: $l_{M'} \geq l_i, i = 1,\ldots,m; i \neq M$. Now, we have the following inequalities for l_M:

$$l_M > 0, \tag{10}$$

$$l_M \geq l_{M'} > l_{M'} - \sum_{\substack{i=1,\\ i\neq M, M'}}^{m} l_i, \tag{11}$$

$$l_M \leq \sum_{\substack{i=1,\\ i\neq M}}^{m} l_i, \tag{12}$$

and therefore $l_M \in A'$, so that according to the induction hypothesis, it can be effectively reached by the chain with $m-1$ links. □

Using Lemma 1, it is now possible to formulate the closure condition for $n \geq 3$. For a given joint k, we define A_k as the interval of lengths, given by Lemma 1, reachable by the chain of the $n-2$ links not adjacent to joint k, i.e., $i \notin \{k-1, k\}$. Then, a *closure condition* for Eq. (1) to have a solution is that the interval I_k defined in Eq. (5) has a non-empty intersection with A_k:

$$I_k \cap A_k \neq \varnothing. \tag{13}$$

Note that an equivalent closure condition can be obtained for each joint k in the linkage, as was the case for the triangle inequalities.

Algorithm 1: *Find Range*

Input : A loop equation e_n in the form of Eq. (1) and a variable θ_k.
Output: The set $\mathcal{S}_k(e_n)$ of feasible values for θ_k.

$I_k \leftarrow [\,|l_{k-1} - l_k|,\, l_{k-1} + l_k]$
$A_{k\,max} \leftarrow \sum_{i=1, i\notin\{k-1,k\}}^{n} l_i$
$l_M \leftarrow \max_{i\notin\{k-1,k\}}\{l_i\}$
$A_{k\,min} \leftarrow \max\{0, l_M - \sum_{i=1, i\notin\{M,k-1,k\}}^{n} l_i\}$
$A_k \leftarrow [A_{k\,min}, A_{k\,max}]$
if $I_k \cap A_k = \varnothing$ **then**
  $\mathcal{S}_k^+ \leftarrow \varnothing$
  $\mathcal{S}_k^- \leftarrow \varnothing$
else
  $[a, b] \leftarrow I_k \cap A_k$
  $\theta_k^+(a) \leftarrow +\arccos((a^2 - l_{k-1}^2 - l_k^2)/2l_{k-1}l_k)$
  $\theta_k^+(b) \leftarrow +\arccos((b^2 - l_{k-1}^2 - l_k^2)/2l_{k-1}l_k)$
  $\mathcal{S}_k^+ \leftarrow [\theta_k^+(b), \theta_k^+(a)]$
  $\mathcal{S}_k^- \leftarrow [2\pi - \theta_k^+(a), 2\pi - \theta_k^+(b)]$

Return: $\mathcal{S}_k(e_n) = \mathcal{S}_k^+ \cup \mathcal{S}_k^-$

When the closure condition holds, the feasibility range $\mathcal{S}_k(e_n)$ for θ_k can be obtained noting that each value in the interval $[a, b] = I_k \cap A_k$ corresponds to a valid length for L_k, which yields two solutions for the angle θ_k in general. By applying the cosine rule to the triangle defined by l_{k-1}, l_k, and L_k (Fig. 1), the corresponding solutions for θ_k, as a function of L_k, are given by

$$\theta_k^{\pm}(L_k) = \pm \arccos\left(\frac{L_k^2 - l_{k-1}^2 - l_k^2}{2l_{k-1}l_k}\right). \tag{14}$$

Thus, the complete set $\mathcal{S}_k(e_n)$ can be easily obtained from the interval $[a, b] = I_k \cap A_k$ of valid values for L_k, simply by computing the angles $\theta_k^+(a)$ and $\theta_k^+(b)$ corresponding to each bound of this interval. Since each determination of the arccosine function is a monotonic continuous function, the set $\mathcal{S}_k(e_n)$ is the union of two equal-length intervals defined by the corresponding bounding values. The complete procedure for computing $\mathcal{S}_k(e_n)$ is given in Algorithm 1.

It is useful to note that the structure of $\mathcal{S}_k(e_n)$ depends on the way A_k and I_k intersect. There are four different possibilities:

1. *A_k and I_k do not intersect.* In this case $\mathcal{S}_k(e_n)$ is **empty**.
2. *A_k and I_k partially overlap.* In this case $\mathcal{S}_k(e_n)$ is **a single interval**, given by $\mathcal{S}_k(e_n) = [\theta_k^+(b), 2\pi - \theta_k^+(b)]$ when A_k contains the lower bound of I_k, and $\mathcal{S}_k(e_n) = [2\pi - \theta_k^+(a), \theta_k^+(a)]$ when A_k contains the upper bound of I_k.
3. *A_k is interior to I_k.* In this case $\mathcal{S}_k(e_n)$ has **two disjoint intervals**, $\mathcal{S}_k^+ = [\theta_k^+(b), \theta_k^+(a)]$ and $\mathcal{S}_k^- = [2\pi - \theta_k^+(a), 2\pi - \theta_k^+(b)]$.
4. *A_k includes I_k.* In this case $\mathcal{S}_k(e_n)$ is **the whole range** $[0, 2\pi]$.

A remark concerning the notation used in this paper is required here: when denoting circular intervals as $[\alpha, \beta]$ we refer to all angular values found between α and β following a counter-clockwise direction. Thus, for example, the interval $[-\pi/2, \pi/2] = [3\pi/2, \pi/2]$ corresponds to the right half of the circumference, while $[\pi/2, -\pi/2] = [\pi/2, 3\pi/2]$ corresponds to the left half. When possible, we will denote angular values by their determination included in $[0, 2\pi]$.

3. Value propagation from one variable to another

Suppose now that we fix some variable θ_j to a given value α, and that we want to obtain the set of values for another variable θ_k that are compatible with such an assignment, i.e., the values for θ_k for which equation e_n with the condition $\theta_j = \alpha$ has a solution. This set will be called the propagation of the value α from θ_j to θ_k in equation e_n, and will be denoted by $\mathcal{P}_{jk}(\alpha)$.

In order to compute $\mathcal{P}_{jk}(\alpha)$, we will first substitute the value α for θ_j in e_n, and we will then find the feasibility range for θ_k in the resulting equation. If we denote the new equation by $(e_n|\theta_j = \alpha)$, and the feasibility range for θ_k in this equation by $\mathcal{S}_k(e_n|\theta_j = \alpha)$, we clearly have $\mathcal{P}_{jk}(\alpha) \equiv \mathcal{S}_k(e_n|\theta_j = \alpha)$.

The substitution $\theta_j = \alpha$ in e_n gives

$$\ldots \mathbf{R}(\theta_{j-1})\ \mathbf{Tx}(l_{j-1})\ \mathbf{R}(\alpha)\ \mathbf{Tx}(l_j)\ \mathbf{R}(\theta_{j+1}) \ldots = \mathbf{I}, \tag{15}$$

but note from Fig. 2 that fixing $\theta_j = \alpha$ corresponds to fixing the distance between joints $j-1$ and $j+1$ to the value

$$l_\alpha = \sqrt{l_{j-1}^2 + l_j^2 + 2 l_{j-1} l_j \cos(\alpha)}. \tag{16}$$

Thus, when $l_\alpha \neq 0$, Eq. (15) can be rewritten as

$$\ldots \mathbf{R}(\tilde{\theta}_{j-1})\ \mathbf{Tx}(l_\alpha)\ \mathbf{R}(\tilde{\theta}_{j+1}) \ldots = \mathbf{I}, \tag{17}$$

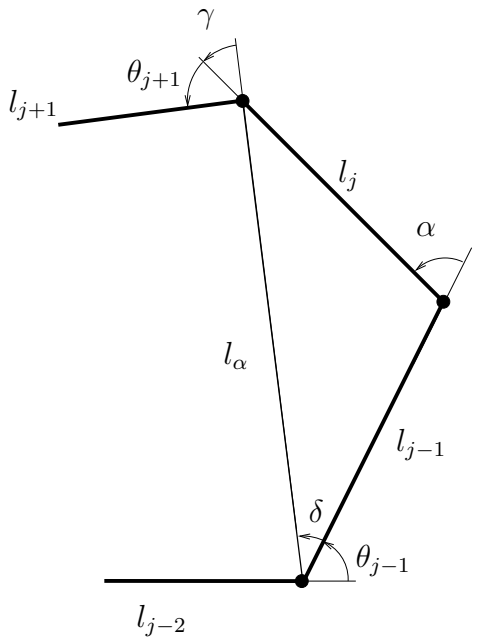


Figure 2: Substitution of variable θ_j by a constant value α.

where

$$\begin{aligned}
\tilde{\theta}_{j-1} &= \theta_{j-1} + \delta, && (18)\\
\tilde{\theta}_{j+1} &= \theta_{j+1} + \gamma, && (19)\\
\delta &= \text{atan2}\left(l_j \sin(\alpha),\, l_{j-1} + l_j \cos(\alpha)\right), && (20)\\
\gamma &= \alpha - \delta, && (21)
\end{aligned}$$

which corresponds to the loop equation of a linkage with $n-1$ links.

When $l_\alpha = 0$, which occurs when $l_j = l_{j-1}$ and $\alpha = \pi$, the linkage is said to be in a *singular configuration*, and Eq. (15) must be rewritten as

$$\ldots\ \mathbf{R}(\tilde{\theta}_{j-1})\ \ldots = \mathbf{I} \tag{22}$$

where

$$\tilde{\theta}_{j-1} = \theta_{j-1} + \pi + \theta_{j+1}, \tag{23}$$

which corresponds to the loop equation of a linkage with $n-2$ links.

Note now that because Eqs. (17) and (22) adopt the standard form of Eq. (1), we can readily use the *Find Range* algorithm to compute the feasibility ranges for their variables. On both equations, the algorithm directly

provides the range $\mathcal{P}_{jk}(\alpha)$ for θ_k if $k \neq j \pm 1$. For $k = j \pm 1$, the *Find Range* algorithm applied on Eq. (17) provides the ranges for $\tilde{\theta}_{j\pm1}$, from which the ranges $\mathcal{P}_{j,j\pm1}(\alpha)$ for $\theta_{j\pm1}$ can be obtained using Eqs. (18)-(21). Applied on Eq. (22), the algorithm provides the range for $\tilde{\theta}_{j-1}$, and depending on whether such range is empty or not, we see from Eq. (23) that the ranges for θ_{j-1} and θ_{j+1} will both be empty or $[0, 2\pi]$.

4. Interval propagation from one variable to another

In some situations, a given variable θ_j may have been constrained to take not just a specific value, but any value inside an interval $C = [\underline{\alpha}, \overline{\alpha}]$. In this case, we are interested in obtaining the set of values for another variable θ_k that are compatible with at least one value $\alpha \in C$ of θ_j, i.e., the set

$$\mathcal{P}_{jk}(C) \equiv \bigcup_{\alpha \in C} \mathcal{P}_{jk}(\alpha), \tag{24}$$

which will be called the propagation of interval C from θ_j to θ_k in equation e_n. In this Section, we will derive an interval propagation algorithm to compute $\mathcal{P}_{jk}(C)$ efficiently. To this end, we next introduce some properties of the propagation map $\mathcal{P}_{jk}(\theta_j)$.

4.1. Properties of the propagation map

Let $\mathcal{P}_{jk}$ denote the graph of $\mathcal{P}_{jk}(\theta_j)$, i.e., the set of all points (α, β) such that $\beta \in \mathcal{P}_{jk}(\alpha)$. Fig. 3 shows an example graph for a particular loop equation, where $\mathcal{P}_{jk}$ is composed of two subsets, $\mathcal{P}^+_{jk}$ and $\mathcal{P}^-_{jk}$, corresponding to the two determinations of the arccosine function, which in this case are disjoint. It is clear that $\beta \in \mathcal{P}_{jk}(\alpha)$ if, and only if, $\alpha \in \mathcal{P}_{kj}(\beta)$, and hence, $(\alpha, \beta) \in \mathcal{P}_{jk} \Leftrightarrow (\beta, \alpha) \in \mathcal{P}_{kj}$, so that the graph of $\mathcal{P}_{jk}(\theta_j)$ can also be interpreted as the graph of the inverse map $\mathcal{P}_{kj}(\theta_k)$ with its axes permuted. In what follows we will say that two variables θ_j and θ_k of equation e_n are *adjacent* if they appear in consecutive joints of the linkage.

Definition 1 (Singular and regular values). Let θ_j and θ_k be two adjacent variables of equation e_n. We say that β is a *singular value* of $\mathcal{P}_{jk}(\theta_j)$ if the substitution $\theta_k = \beta$ in e_n gives rise to a singular configuration of the linkage as defined in Section 3. Points $(\alpha, \beta) \in \mathcal{P}_{jk}$ for which β is a singular value of $\mathcal{P}_{jk}(\theta_j)$ will be called *singular points* of $\mathcal{P}_{jk}$. Non-singular values of $\mathcal{P}_{jk}(\theta_j)$ will be called *regular values* of $\mathcal{P}_{jk}(\theta_j)$, and points $(\alpha, \beta) \in \mathcal{P}_{jk}$ for which β is a regular value of $\mathcal{P}_{jk}(\theta_j)$ will be called *regular points* of $\mathcal{P}_{jk}$.

To avoid confusions, we remark that singular values of $\mathcal{P}_{jk}(\theta_j)$ are values taken by θ_k, not by θ_j. Singular values always correspond to $\beta = \pi$, and they appear when the two links joined by θ_k have the same length, in which case, $\beta = \pi$ is a singular value of the two maps $\mathcal{P}_{jk}(\theta_j)$ with $j = k \pm 1$. Also observe that, if $(\alpha, \beta) \in \mathcal{P}_{jk}$ is a singular point of $\mathcal{P}_{jk}$, then $(\theta_j, \beta) \in \mathcal{P}_{jk}$ for all θ_j, and all of them are singular points, and $\mathcal{P}_{kj}(\beta) = [0, 2\pi]$ or, equivalently, $\beta \in \mathcal{P}_{jk}(\theta_j)$ for all $\theta_j \in [0, 2\pi]$.

It is important to note that the bounds of the intervals forming the image set $\mathcal{P}_{jk}(\alpha)$ vary smoothly with α except, possibly, at singular values of the

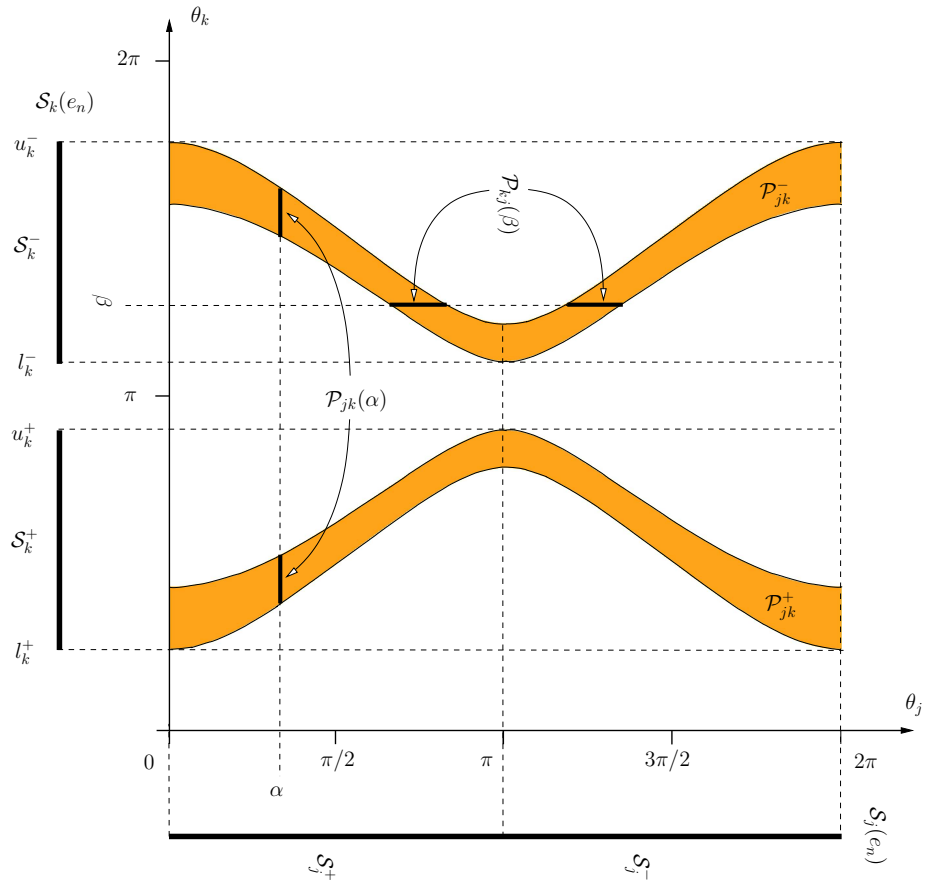


Figure 3: Graph of $\mathcal{P}_{jk}(\theta_j)$ for variables $\theta_j = \theta_2$ and $\theta_k = \theta_5$ in the loop equation $\mathbf{R}(\theta_1)\mathbf{Tx}(4)\mathbf{R}(\theta_2)\mathbf{Tx}(8)\mathbf{R}(\theta_3)\mathbf{Tx}(1)\mathbf{R}(\theta_4)\mathbf{Tx}(8)\mathbf{R}(\theta_5)\mathbf{Tx}(6) = \mathbf{I}$. The same figure can also be interpreted as the graph of $\mathcal{P}_{kj}(\theta_k)$ by looking at it with a left turn of 90°. The slices $\theta_j = \alpha$ and $\theta_k = \beta$ provide $\mathcal{P}_{jk}(\alpha)$ and $\mathcal{P}_{kj}(\beta)$, respectively.

inverse map $\mathcal{P}_{kj}(\theta_k)$. Effectively, to compute the set $\mathcal{P}_{jk}(\alpha) = \mathcal{S}_k(e_n|\theta_j = \alpha)$ when α is a regular value of $\mathcal{P}_{kj}(\theta_k)$, first, equation $(e_n|\theta_j = \alpha)$ will be rewritten using Eqs. (16)-(21), then, using the *Find Range* algorithm, the corresponding intervals I_k and A_k, defined as in (13), will be determined and their intersection found, and finally, if this intersection is not empty, the bounds for θ_k will be obtained using (14). All of the functions involved in the process are continuous with respect to α, with the only exception of (20), which becomes undetermined just when α is a singular value of $\mathcal{P}_{kj}(\theta_k)$. Therefore, while θ_j takes only regular values of $\mathcal{P}_{kj}(\theta_k)$, the bounds of $\mathcal{P}_{jk}(\theta_j)$ vary continuously with θ_j, eventually splitting into two equal length intervals (or joining into a single one), or perhaps shrinking to a point and then vanishing when the intersection $I_k \cap A_k$ becomes empty (or growing from a single initial point when I_k and A_k enter in contact).

On the contrary, if a value $\alpha \in \mathcal{S}_j(e_n)$ is a singular value of $\mathcal{P}_{kj}(\theta_k)$, then $\mathcal{P}_{jk}(\alpha) = [0, 2\pi]$, as discussed above, and it may correspond to a discontinuity in the bounds of $\mathcal{P}_{jk}(\theta_j)$.

Our purpose, now, is to show that those local extrema of $\mathcal{P}_{jk}(\theta_j)$ that correspond to regular values of θ_k must coincide with one of the bounds that define the set $\mathcal{S}_k(e_n)$ of valid values for θ_k. But before, since $\mathcal{P}_{jk}(\theta_j)$ is not a function (the image of each point is composed, in general, of one or two intervals), we must introduce an appropriate definition of local extremum of $\mathcal{P}_{jk}$.

Definition 2 (Local extrema). A point $(\alpha_m, \beta_m) \in \mathcal{P}_{jk}$ is a *local maximum* of $\mathcal{P}_{jk}$ if there exists a ball B_ρ with center (α_m, β_m) and radius $\rho > 0$ such that $(\alpha, \beta) \in \mathcal{P}_{jk} \cap B_\rho \Rightarrow \beta \in [\beta_m - \rho, \beta_m]$. Similarly, (α_m, β_m) is a *local minimum* of $\mathcal{P}_{jk}$ if $(\alpha, \beta) \in \mathcal{P}_{jk} \cap B_\rho \Rightarrow \beta \in [\beta_m, \beta_m + \rho]$. If (α_m, β_m) is a local maximum (resp. minimum) of $\mathcal{P}_{jk}$, we will say that β_m is a local maximum (resp. minimum) *value* of $\mathcal{P}_{jk}(\theta_j)$. A local maximum or minimum will be called *strict* if (α_m, β_m) is the only point $(\alpha, \beta) \in \mathcal{P}_{jk} \cap B_\rho$ such that $\beta = \beta_m$.

With these definitions, we can see that, in the example of Fig. 3, $\mathcal{P}_{jk}$ has two strict local maxima, $(0, u_k^-)$ and (π, u_k^+), and two strict local minima, $(0, l_k^+)$ and (π, l_k^-). On the contrary, $\mathcal{P}_{kj}(\theta_k)$ has no local extrema.

There is an important relationship between strict local extrema and regular points: If (α_m, β_m) is a local extremum of $\mathcal{P}_{jk}$, the inverse map $\mathcal{P}_{kj}(\theta_k)$ changes from empty to non-empty (or vice-versa) when θ_k passes through

β_m. Whether (α_m, β_m) is a strict local extremum or not depends on whether $\mathcal{P}_{kj}(\beta_m)$ is a single point or an interval. As we have seen previously in this Section, if (α_m, β_m) is a singular point of $\mathcal{P}_{jk}$, then $\mathcal{P}_{kj}(\beta_m) = [0, 2\pi]$, while if it is regular, $\mathcal{P}_{kj}(\beta_m)$ must reduce to a point before vanishing. Thus, we can conclude that *a local extremum of $\mathcal{P}_{jk}$ is strict if, and only if, it is a regular point of $\mathcal{P}_{jk}$*. Next we prove the main property of $\mathcal{P}_{jk}$ we were seeking.

Property 1. *If (α_m, β_m) is a strict local extremum of $\mathcal{P}_{jk}$, then β_m is a bound of one of the intervals composing the feasibility range $\mathcal{S}_k(e_n)$ for θ_k.*

Proof. We prove the property for local maxima; the proof for minima is analogous. The condition for a value β to be the upper bound of an interval of $\mathcal{S}_k(e_n)$ is that $\mathcal{P}_{kj}(\beta) \neq \varnothing$ and $\mathcal{P}_{kj}(\beta + \varepsilon) = \varnothing$ for any sufficiently small $\varepsilon > 0$. Therefore, it is enough to show that if (α_m, β_m) is a strict local maximum of $\mathcal{P}_{jk}$, then for any sufficiently small $\varepsilon > 0$, $\mathcal{P}_{kj}(\beta_m + \varepsilon) = \varnothing$.

The proof is by contradiction. Let us assume that (α_m, β_m) is a strict local maximum of $\mathcal{P}_{jk}$, but $\mathcal{P}_{kj}(\beta_m + \varepsilon) \neq \varnothing$ for arbitrarily small $\varepsilon > 0$. This means that we can find values α_ε such that, for arbitrarily small $\varepsilon > 0$, $(\alpha_\varepsilon, \beta_m + \varepsilon) \in \mathcal{P}_{jk}$. But, since (α_m, β_m) is a local maximum of $\mathcal{P}_{jk}$, there is a ball B_ρ centered at (α_m, β_m) with radius $\rho > 0$ to which points $(\alpha_\varepsilon, \beta_m + \varepsilon)$ can not belong, what means that $|\alpha_\varepsilon - \alpha_m| > \rho$. This implies that there is a discontinuity in the bounds of $\mathcal{P}_{kj}(\theta_k)$ at $\theta_k = \beta_m$, which is only possible if β_m is a singular value of $\mathcal{P}_{jk}(\theta_j)$, in which case, (α_m, β_m) could not be a strict local maximum of $\mathcal{P}_{jk}$. □

4.2. Interval propagation algorithm

In order to determine the set $\mathcal{P}_{jk}(C)$ for $C = [\underline{\alpha}, \overline{\alpha}]$, we already know that

$$\{\mathcal{P}_{jk}(\underline{\alpha}) \cup \mathcal{P}_{jk}(\overline{\alpha})\} \subset \mathcal{P}_{jk}(C) \subset \mathcal{S}_k(e_n). \tag{25}$$

Thus, we are led to determine which parts of the difference set

$$\mathcal{D}_{jk}(C) \equiv \mathcal{S}_k(e_n) \setminus \{\mathcal{P}_{jk}(\underline{\alpha}) \cup \mathcal{P}_{jk}(\overline{\alpha})\} \tag{26}$$

are also included in $\mathcal{P}_{jk}(C)$ (Fig. 4). For this we will use an important fact about $\mathcal{D}_{jk}(C)$, derived from Property 1 of $\mathcal{P}_{jk}$:

Property 2. *If D is an interval for θ_k fully contained in the difference set $\mathcal{D}_{jk}(C)$, then either $D \subset \mathcal{P}_{jk}(C)$, or $D \cap \mathcal{P}_{jk}(C) = \varnothing$.*

PROOF. We first show that if D contains a local extremum of $\mathcal{P}_{jk}(\theta_j)$, then it must be a strict local extremum. To show this, observe that D can not contain singular values of $\mathcal{P}_{jk}(\theta_j)$ because, if $\beta \in \mathcal{S}_k(e_n)$ is a singular value of $\mathcal{P}_{jk}(\theta_j)$, then, as discussed before, $\beta \in \mathcal{P}_{jk}(\alpha)$ for all α, so that, according to (26), β would be excluded from $\mathcal{D}_{jk}(C)$, and hence from D. Since we have already established that a local extremum is strict if and only if it is regular, we conclude that D can only contain strict local extrema.

Now, to prove the property, assume that there are two points, $\varphi \in D$ and $\xi \in D$, such that $\varphi \in \mathcal{P}_{jk}(C)$ and $\xi \notin \mathcal{P}_{jk}(C)$. We will see that this yields a contradiction. Let $\underline{d}$ and $\overline{d}$ be, respectively, the lower and upper bounds of D (which may be included in D, or not). Assume by the moment that $\varphi \in [\underline{d}, \xi]$. Let Ω be the maximum value in $[\varphi, \xi]$ such that $\Omega \in \mathcal{P}_{jk}(C)$. This

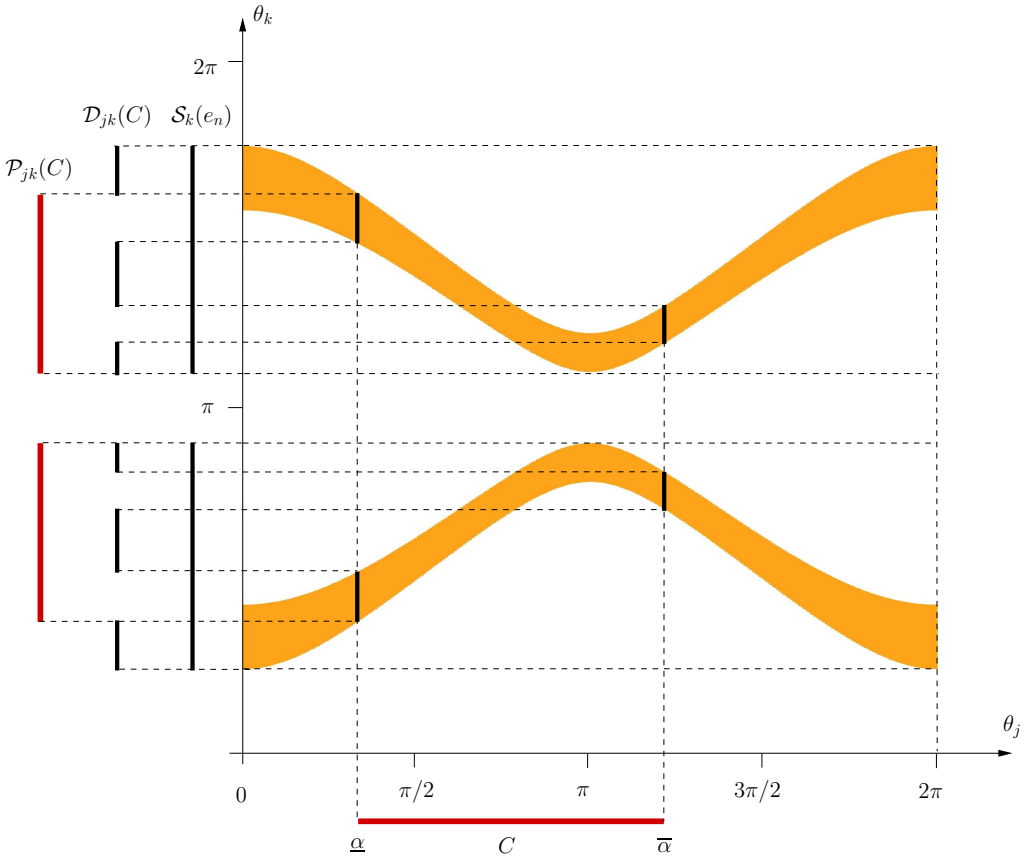


Figure 4: Propagation of interval $C = [\underline{\alpha}, \overline{\alpha}]$ from θ_j to θ_k, obtaining $\mathcal{P}_{jk}(C)$.

implies that $\xi \notin [\underline{d}, \Omega]$. Since Ω is a maximum value of $\mathcal{P}_{jk}(C) = \mathcal{P}_{jk}([\underline{\alpha}, \overline{\alpha}])$, and $\Omega \notin \{\mathcal{P}_{jk}(\overline{\alpha}) \cup \mathcal{P}_{jk}(\underline{\alpha})\}$, this means that Ω is a local maximum value of $\mathcal{P}_{jk}(\theta_j)$. Since $\Omega \in D$, it is a strict local maximum of $\mathcal{P}_{jk}(\theta_j)$ and, by Property 1, Ω must be an upper bound of $\mathcal{S}_k(e_n)$. By construction of $\mathcal{D}_{jk}(C)$, if D contains an upper bound of $\mathcal{S}_k(e_n)$, it must be the upper bound of D, so that we have $\overline{d} = \Omega$, and therefore $[\underline{d}, \Omega] = [\underline{d}, \overline{d}] \supset D$, and the above implication $\xi \notin [\underline{d}, \Omega]$ would mean that $\xi \notin D$, in contradiction with the hypothesis. A similar reasoning can be done assuming that $\varphi \in [\xi, \overline{d}]$. $\square$

Property 2 shows that if D is an interval contained in $\mathcal{D}_{jk}(C)$, then it must be either completely contained in $\mathcal{P}_{jk}(C)$ or disjoint with it, so that it is enough to check the inclusion in $\mathcal{P}_{jk}(C)$ of a representative point $\xi_k \in D$ to determine whether the whole interval D is included in, or excluded from, $\mathcal{P}_{jk}(C)$. The inclusion of a point ξ_k into $\mathcal{P}_{jk}(C)$ can be checked by computing $\mathcal{P}_{kj}(\xi_k)$ and intersecting it with C. The intersection will be non-empty if, and only if, C contains some $\xi_j \in \mathcal{P}_{kj}(\xi_k)$, which is equivalent to saying that $\xi_k \in \mathcal{P}_{jk}(\xi_j) \subset \mathcal{P}_{jk}(C)$. Then, since $\mathcal{D}_{jk}(C)$ can be decomposed into a collection of intervals D_i, we can determine which parts of $\mathcal{D}_{jk}(C)$ are included in $\mathcal{P}_{jk}(C)$ by checking the inclusion of a single point of each D_i (e.g., its middle point).

Summarizing, the algorithm for the propagation of the interval C from θ_j to θ_k (Algorithm 2) requires finding the feasibility range $\mathcal{S}_k(e_n)$ using the algorithm *Find Range* presented in Section 2, propagating the two bounds of C to θ_k as explained in Section 3, and, eventually, propagating an interior point of each connected interval of the difference set $\mathcal{D}_{jk}(C)$ to θ_j, which in the worst case will consist of six different intervals.

5. Simultaneous propagation of intervals for multiple variables

In this Section, we consider the general case in which constraints are imposed on several variables at the same time, and the goal is to find the set of values for another variable that are compatible with all these constraints simultaneously.

In the simplest case, which we will call multiple value propagation, each constrained variable is restricted to a single value. In this case, if $\theta_1, \ldots, \theta_r$[1]

[1]Without loss of generality, and to ease the notation, we rename the constrained variables with consecutive indexes 1 to r.

Algorithm 2: *Propagate Interval*

Input : A loop equation e_n, an input variable θ_j, its interval constraint $C = [\underline{\alpha}, \overline{\alpha}]$, and an output variable θ_k.
Output: The set $\mathcal{P}_{jk}(C)$ of compatible values for θ_k.

$S \leftarrow \mathcal{S}_k(e_n)$ // Call the algorithm *Find Range*
$P \leftarrow \mathcal{P}_{jk}(\underline{\alpha}) \cup \mathcal{P}_{jk}(\overline{\alpha})$ // Propagate $\underline{\alpha}$ and $\overline{\alpha}$ (Section 3)
$D \leftarrow S \setminus P$ // Compute the difference set $\mathcal{D}_{jk}(C)$
foreach *connected interval* D_i *forming* D **do**
  $\xi_k \leftarrow$ midpoint of D_i
  $P_j \leftarrow \mathcal{P}_{kj}(\xi_k)$ // Propagate the value ξ_k (Section 3)
  if $P_j \cap C \neq \varnothing$ **then**
    $P \leftarrow P \cup D_i$

Return: $\mathcal{P}_{jk}(C) = P$

are respectively restricted to the values $\alpha_1, \ldots, \alpha_r$, the set of values for θ_k simultaneously compatible with all these constraints will be denoted by $\mathcal{P}_{1\ldots r,k}(\alpha_1, \ldots, \alpha_r)$. This set can be easily computed by recursively applying the method for equation rewriting used for value propagation (Section 3). The process starts by fixing θ_1 to the value α_1 and rewriting the loop equation e_n to get $(e_n|\theta_1 = \alpha_1)$, then fixing θ_2 to the value α_2 in the new equation to get $(e_n|\theta_1 = \alpha_1, \theta_2 = \alpha_2)$, and so on, until all constrained variables are fixed, giving rise to the equation $(e_n|\theta_1 = \alpha_1, \ldots, \theta_r = \alpha_r)$. Then, the set of compatible values for θ_k can be obtained using the algorithm *Find Range* with this equation, taking into account the possible changes of variables affecting θ_k that may have been introduced along the process.

In the general case, different variables may have been constrained with either interval or value constraints, in which case the above procedure is not enough, since the method of equation rewriting can not be applied with interval constraints. Thus, if p variables $\theta_1, \ldots, \theta_p$ are respectively constrained to the intervals $C_1, \ldots, C_p$, and $r - p$ variables $\theta_{p+1}, \ldots, \theta_r$ are constrained to the $r - p$ values $\alpha_{p+1}, \ldots, \alpha_r$, we are interested in obtaining the set $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_p, \alpha_{p+1}, \ldots, \alpha_r)$ of values for θ_k that are compatible with all of these constraints simultaneously. In this case, we can use the rewriting method with the $r - p$ value constraints, reducing the problem to one with

only interval constraints. We will use the notation

$$\mathcal{P}_{1\ldots p,k}(C_1 \ldots C_p | \theta_{p+1} = \alpha_{p+1} \ldots, \theta_r = \alpha_r) \equiv \mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_p, \alpha_{p+1}, \ldots, \alpha_r),$$

to indicate that the result of propagating multiple value and interval constrains in equation e_n is equivalent to propagating only the interval constraints in equation $(e_n | \theta_{p+1} = \alpha_{p+1}, \ldots, \theta_r = \alpha_r)$.

Now, to derive an algorithm for multiple interval propagation, we consider the properties of the multiple propagation map $\mathcal{P}_{1\ldots r,k}(\theta_1, \ldots, \theta_r)$ and its graph $\mathcal{P}_{1\ldots r,k}$, whose local extrema are defined in a way completely analogous to that of Definition 2. We next show that, like in the $r = 1$ case, the sets $\mathcal{P}_{1\ldots r,k}$ satisfy the important property that all of their strict local extrema correspond to bounds of $\mathcal{S}_k(e_n)$.

Property 3 (Generalization of Property 1). *If a point $(\alpha_1, \ldots, \alpha_r, \beta_m)$ is a strict local extremum of $\mathcal{P}_{1\ldots r,k}$, then β_m is a bound of one of the intervals composing the feasibility range $\mathcal{S}_k(e_n)$ for θ_k.*

Proof. We prove the property for the case $r = 2$ corresponding to $\mathcal{P}_{ijk}$. The extension to higher dimensions is easily obtained by induction on the number r of input variables.

Consider the set $\mathcal{P}_{ijk}$ for a given equation e_n (see Fig. 5 for a particular example). Note that its projection onto the θ_j-θ_k plane is the graph of $\mathcal{P}_{jk}(\theta_j)$. Let $P = (\zeta_m, \alpha_m, \beta_m)$ be a strict local maximum of $\mathcal{P}_{ijk}$. Consider the slice of $\mathcal{P}_{ijk}$ normal to the θ_j axis, taken at $\theta_j = \alpha_m$. This slice is actually the graph of $\mathcal{P}_{ik}(\theta_i | \theta_j = \alpha_m)$, and its projection onto the θ_j-θ_k plane is $\mathcal{P}_{jk}(\alpha_m)$. Since $P = (\zeta_m, \alpha_m, \beta_m)$ is a strict local maximum of $\mathcal{P}_{ijk}$, then (ζ_m, β_m) is a strict local maximum of the slice $\mathcal{P}_{ik}(\theta_i | \theta_j = \alpha_m)$ containing P. By applying Property 1 on the graph of the map $\mathcal{P}_{ik}(\theta_i | \theta_j = \alpha_m)$, β_m must be a bound of $\mathcal{S}_k(e_n | \theta_j = \alpha_m) = \mathcal{P}_{jk}(\alpha_m)$. Since P is a strict local maximum of $\mathcal{P}_{ijk}$, this bound must decrease for slices of $\mathcal{P}_{ijk}$ taken at values slightly different from $\theta_j = \alpha_m$, which means that (α_m, β_m) must also be a strict local maximum of the projected graph $\mathcal{P}_{jk}$. Now, using Property 1 on $\mathcal{P}_{jk}$, we conclude that β_m must be a bound of $\mathcal{S}_k(e_n)$. □

For the generalization of Property 2 we first consider the case $r = 2$. In this case, we are interested in $\mathcal{P}_{ijk}(C_i, C_j)$ and, if $C_i = [\underline{\alpha}_i, \overline{\alpha}_i]$, we can say that

$$\{\mathcal{P}_{ijk}(\underline{\alpha}_i, C_j) \cup \mathcal{P}_{ijk}(\overline{\alpha}_i, C_j)\} \subset \mathcal{P}_{ijk}(C_i, C_j) \subset \mathcal{S}_k(e_n). \tag{27}$$

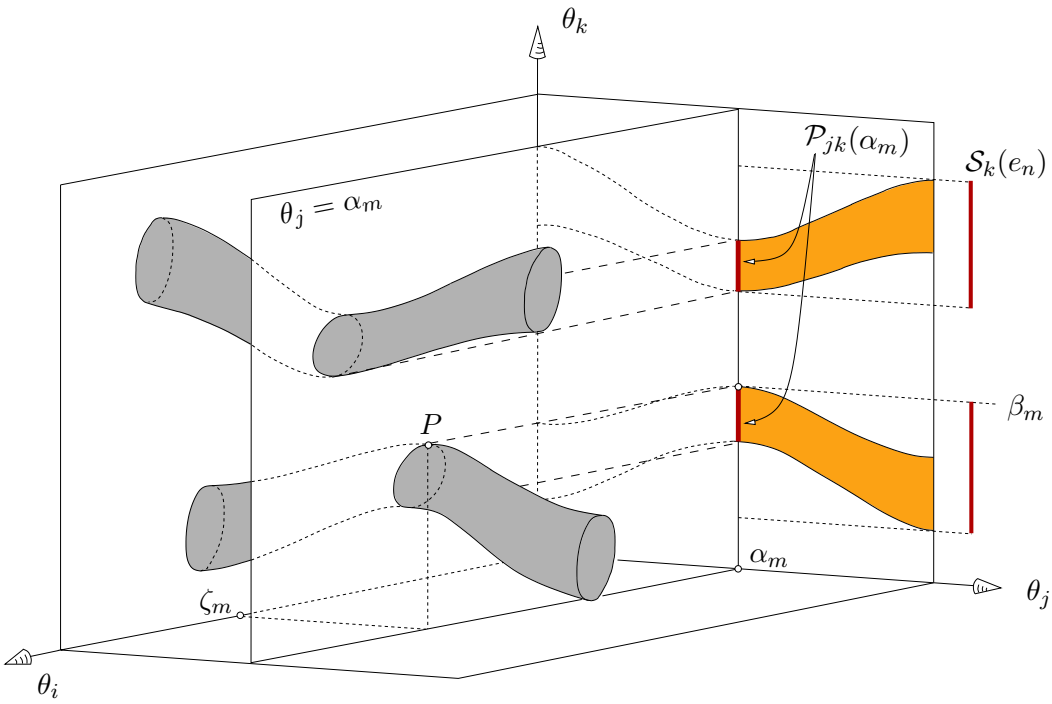


Figure 5: The graph of an example map $\mathcal{P}_{ijk}(\theta_i, \theta_j)$ and its projection onto the θ_j-θ_k plane, $\mathcal{P}_{jk}(\theta_j)$.

Similar inclusions can be established using the bounds of $C_j = [\underline{\alpha}_j, \overline{\alpha}_j]$, so that we can say that the set of values of θ_k compatible with some point in the box $C_i \times C_j$ of the space of θ_i-θ_j obviously includes the set of values of θ_k compatible with the points lying in any of the four edges of the box. Now, we can construct the difference set

$$\mathcal{D}_{ijk}(C_i, C_j) = \mathcal{S}_k(e_n) \backslash \{\mathcal{P}_{ijk}(\underline{\alpha}_i, C_j) \cup \mathcal{P}_{ijk}(\overline{\alpha}_i, C_j) \cup \mathcal{P}_{ijk}(C_i, \underline{\alpha}_j) \cup \mathcal{P}_{ijk}(C_i, \overline{\alpha}_j)\},$$

and, as in the $r = 1$ case, if $\beta_k \in \mathcal{D}_{ijk}(C_i, C_j)$, it can not be a non-strict local extreme value of $\mathcal{P}_{ijk}(\theta_i, \theta_j)$. Otherwise, at least one of the projections of β_k onto the θ_i-θ_k or θ_j-θ_k planes would be a non-strict local extreme value of the corresponding map, $\mathcal{P}_{ik}(\theta_i)$ or $\mathcal{P}_{jk}(\theta_j)$, and consequently, β_k would be a singular value of one of these maps. This would imply that $\beta_k \in \mathcal{P}_{ik}(\underline{\alpha}_i)$ or $\beta_k \in \mathcal{P}_{jk}(\underline{\alpha}_j)$, and would be excluded from $\mathcal{D}_{ijk}(C_i, C_j)$. Therefore, we can use Property 3 to conclude, following a reasoning analogous to that of Property 2, that each connected interval of $\mathcal{D}_{ijk}(C_i, C_j)$ must either be

completely included in, or completely disjoint with $\mathcal{P}_{ijk}(C_i, C_j)$.

In the general case, for $r \geq 2$, the difference set $\mathcal{D}_{1\ldots r,k}(C_1, \ldots, C_r)$ is defined by subtracting from $\mathcal{S}_k(e_n)$ all the $(r-1)$-dimensional walls of the box $C_1 \times \ldots \times C_r$. It is not hard to see that the intervals composing the difference set $\mathcal{D}_{1\ldots r,k}(C_1, \ldots, C_r)$ defined in this way always satisfy the property of being either completely included in, or completely disjoint with the set $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_r)$, so that it is enough to check the inclusion in this set of a single point in each connected interval forming $\mathcal{D}_{1\ldots r,k}(C_1, \ldots, C_r)$, to fully determine the complete set $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_r)$.

Algorithm 3 summarizes the multiple interval propagation procedure. Note that the algorithm is defined recursively, since the propagation of a box of dimension r requires $2r$ calls to the algorithm to propagate its $2r$ walls, which, in turn, are boxes of dimension $r-1$. To this end we use the equivalence:

$$\mathcal{P}_{1\ldots r,k}(C_1, \ldots, \alpha_i, \ldots, C_r) \equiv \mathcal{P}_{1\ldots i-1,i+1\ldots r,k}(C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_r | \theta_i = \alpha_i),$$

which is computed by calling the algorithm with one less argument, but using equation $(e_n|\theta_i = \alpha_i)$ instead of e_n.

To check the inclusion of a value ξ_k of θ_k in $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_r)$, we fix this value in e_n to get $(e_n|\theta_k = \xi_k)$ and check the compatibility of the intervals C_i for this equation. This can be achieved by intersecting, e.g., C_1 with the propagation of the $r-1$ remaining intervals into θ_1: $\mathcal{P}_{2\ldots r,1}(C_2 \ldots C_r|\theta_k = \xi_k)$. If this intersection is not empty, the intervals are compatible and the interval D_i containing ξ_k must be included in the result.

6. Solving multi-loop linkages

In the preceding sections we dealt only with single-loop linkages. General linkages, though, may consist of multiple interconnected loops sharing some joints between them (Fig. 6). In this case, one loop equation is imposed by each one of the loops, and a value for a variable, to be feasible, must be compatible, not only with a solution of the equations involving this variable, but with a global solution for all variables of all loop equations simultaneously. Unfortunately, no procedure is known to compute the exact feasibility ranges for variables in general multi-loop linkages but, as shown next, the propagation algorithm of the previous section can be used iteratively to isolate all the feasible configurations of such linkages.

Algorithm 3: *Propagate Multiple Intervals*

Input : A loop equation e_n, r input variables $\theta_1, \ldots, \theta_r$, their respective interval constraints $C_1, \ldots, C_r$, and an output variable θ_k.

Output: The set $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_r)$ of compatible values for θ_k.

if $r = 1$ **then**
  $P \leftarrow \mathcal{P}_{1k}(C_1)$ `// Call the algorithm` ***Propagate Interval***
else
  for $i = 1, \ldots, r$ **do** `// Propagate all (`r`-1)-dimensional walls`
    `// Recursive call using the lower bound`
    $\underline{\alpha}_i \leftarrow$ lower bound of C_i
    $L_i \leftarrow \mathcal{P}_{1\ldots i-1,i+1\ldots r,k}(C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_r | \theta_i = \underline{\alpha}_i)$
    `// Recursive call using the upper bound`
    $\overline{\alpha}_i \leftarrow$ upper bound of C_i
    $U_i \leftarrow \mathcal{P}_{1\ldots i-1,i+1\ldots r,k}(C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_r | \theta_i = \overline{\alpha}_i)$
  $P \leftarrow \bigcup_{i=1}^{r} \{L_i \cup U_i\}$
  $S \leftarrow \mathcal{S}_k(e_n)$ `// Call the algorithm` ***Find Range***
  $D \leftarrow S \setminus P$ `// Compute the difference set` $\mathcal{D}_{1\ldots r,k}(C_1, \ldots, C_r)$
  foreach *connected interval* D_i *forming* D **do**
    $\xi_k \leftarrow$ midpoint of D_i
    $P_1 \leftarrow \mathcal{P}_{2\ldots r,1}(C_2, \ldots, C_r | \theta_k = \xi_k)$ `// Propagation to` θ_1
    if $P_1 \cap C_1 \neq \varnothing$ **then**
      $P \leftarrow P \cup D_i$

Return: $\mathcal{P}_{1\ldots r,k}(C_1, \ldots, C_r) = P$

To this end, we start choosing a loop basis of the linkage [11] and writing an equation of the form of Eq. (1) for each loop in the basis. Note that the longer a loop is, the less constrained its angles result. Thus, among all possible loop bases, the *minimal* ones are preferred in practice [12], as they keep the loop lengths to a minimum. Then, we label as *shared* those variables corresponding to joints involved in more than one loop. Let s be the number of shared variables and consider an s-dimensional box B defined as the Cartesian product of given initial intervals for the shared variables in which solutions are to be sought, e.g., $B = [0, 2\pi]^s$ if the whole set of solutions

is wanted. For each equation, we (1) get the list of shared variables involved in the equation and their corresponding intervals defining the box, (2) for each variable in the list, use Algorithm 3 to compute the set of values for the variable that are compatible with the intervals of the remaining variables in the equation, intersect the result with the interval for the variable, and replace it in the definition of the box, and (3) in the case that, as a result of the previous step, the interval for some variable gets split into two or more intervals, take the fist interval as the new one for the box and create, for each remaining interval, a new "child" box that will be included in a list of boxes to be identically processed later. Any time the interval of a shared variable is reduced as a consequence of this process, the variable is said to have been "touched". The touching of a variable triggers all equations sharing the variable (except the one that provoked the touch), and all triggered equations are kept on a "wake-up" stack. To ensure that all possible propagations are performed, we repeatedly select one equation from the stack, apply steps (1) to (3) to the equation, and if some variable becomes touched, we trigger any other equations sharing the variable, until the wake-up stack gets empty. The latter situation is known as a *fixpoint* of the propagation process [13], meaning that all intervals for the shared variables defining the box are locally consistent within each equation. However, this does not necessarily mean that these intervals are globally consistent, in the sense that we can not guarantee that any value in the interval of a variable is compatible with a feasible configuration of the linkage.

To isolate all feasible configurations, we embed the previous process in a standard branch-and-prune algorithm [14]. Two operators will be employed to this end, Trim-Box and Split-Box. The former prunes portions of a box containing no solution, by executing a single iteration of the propagation mechanism of the previous paragraph. The latter simply bisects a box into two sub-boxes by dividing its largest interval at its midpoint. Initially, the algorithm applies Trim-Box to the whole box $[0, 2\pi]^s$. As mentioned earlier, such reduction may yield several compatible child boxes in general, and either one of four actions is applied on each one of them:

1. If the box has an empty interval, then it contains no solution and the box is labeled as *empty*.
2. If the box is "small enough", then the box is labeled as a *solution* box.
3. If the box is not "small enough" and a fixpoint has not been reached, then Trim-Box is applied.

4. If the box is not "small enough" and a fixpoint has been reached, then the box is bisected into two sub-boxes using SPLIT-BOX.

If the last case occurs, the whole process is recursively applied to the new sub-boxes until all non-empty boxes are "small enough". A small-enough box is defined as one for which all of its intervals are shorter than a given threshold σ.

Note that, on termination, this process will have explored a tree of boxes whose internal nodes are boxes being split at some time, and whose leaves are either solution or empty boxes. Solution boxes form an approximation of the set of mutually compatible values for the shared variables that correspond to feasible configurations of the linkage. Actual linkage configurations can be easily obtained from each one of the solutions found for the shared variables by propagation to the non-shared variables. Note also that the accuracy of the box approximation can be adjusted through the σ threshold. In other words, the lower the chosen σ, the less the error committed when approximating a solution by any point inside a box.

7. Experiments

The previous algorithm has been implemented in C++, and all CPU times will be given for an Intel(R) Core(TM) i7 870 processor at 2.93 GHz, under Windows 7.

Results for two experiments are provided. The first one solves the position analysis of the "double butterfly" linkage (Fig. 6) when θ_3 is a fixed, known angle, which yields a finite number of isolated solutions. The second one solves the same linkage but assuming that θ_3 is a free variable, yielding a one-dimensional continuum of solutions. The same benchmarks have been used previously to show the performance of elimination [15, 16], continuation [17, 18], and relaxation techniques [2]. We compare our results with those derived by such techniques, and employ the same linkage dimensions used in these

papers. Namely:

$$\begin{array}{llll} a_0 = 7 & b_0 = 13 & \gamma_0 = 36.87^\circ & \\ a_1 = 7 & b_1 = 6 & \gamma_1 = 22.62^\circ & \\ a_2 = 5 & b_2 = 3 & \gamma_2 = 53.13^\circ & \\ a_6 = 3 & b_6 = 2 & \gamma_6 = 36.87^\circ & \\ l_3 = 7 & l_4 = 9 & l_5 = 12 & l_7 = 11 \end{array}$$

7.1. A rigid butterfly

The number of solutions of the double butterfly linkage varies depending on which joint angle is fixed and on the specific value given to it. If we set $\theta_3 = 75.75^\circ$, the number of obtained solutions is six [2, 15, 16]. They are given in Table 1. Actually the cited papers employ absolute orientation angles for the links. Fixing our $\theta_3 = 75.75^\circ$ corresponds to fixing the angle θ_6 used

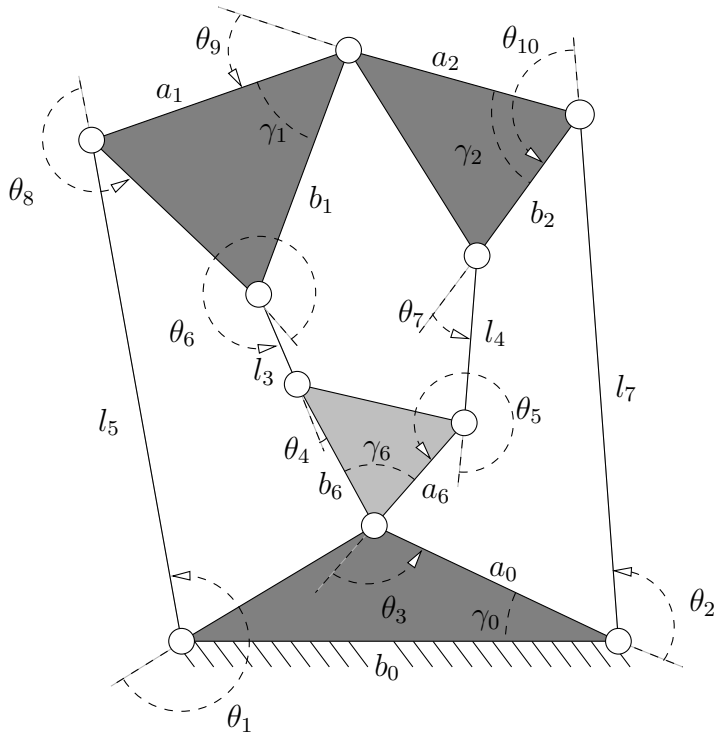


Figure 6: The double butterfly linkage involves three interconnected loops.

in those works to 67.38°. We note that, while continuation and elimination methods must filter the solutions among the eighteen possible complex roots, the method given here directly provides the six real solutions shown in the table. The obtained solutions are in accordance with those in [2, 15, 16].

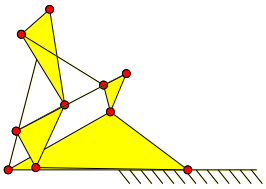
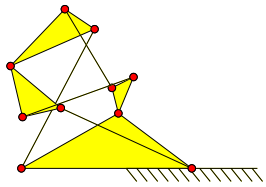
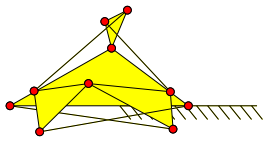
Due to the nature of the algorithm all solutions are obtained as intervals that bound them, which allows estimating the error with respect to the exact position of the roots. This error must be equal or less than the chosen σ threshold, which was set to 10^{-4} in this case. The solutions were found in 0.25 sec of CPU time, after processing 11 boxes. From them, only the six shown in Table 1 were labeled as solutions (thus returning the minimum possible number of boxes) and 5 boxes were found to be empty.

It is difficult to tell at this point whether the presented algorithm outperforms the previous methods based on Dixon's resultant [15, 16], mainly because no statistics are given in this respect in those works, and we have found no publicly available package implementing them. We have checked, however, that our method converges in substantially shorter times than those used by the continuation method in [17, 18], using the implementation available at Jan Verschelde's home page, which spent about 3 seconds of CPU time on the same example. We remark, though, that we are comparing our algorithm with a general-purpose solver targeted to arbitrary systems of algebraic equations, and that a better performance of our algorithm was to be expected, given that we exploit the specific structure of the equations involved. Moreover, in the experiments done on this and other rigid linkages of a similar size, the algorithm converges at rates similar to those of relaxation methods [2].

7.2. A mobile butterfly

If we now free θ_3, a one-dimensional continuum of solutions is obtained. Fig. 7 depicts the projection of the returned boxes onto the θ_{10}-θ_5 plane, on four successive runs of the algorithm, at decreasing values of the σ parameter. If the algorithm is exploring in breadth-first order, the first three plots can also be interpreted as earlier stages of the run for the fourth case ($\sigma = 0.005$). In every plot we indicate the σ threshold, the CPU time spent in seconds (t), the number of solution boxes returned (n_s), and the number of empty boxes found (n_e).

We note that, although from the plots it seems that the different solution branches cross at many points, these are not true bifurcations of the linkage, as revealed by observing other 2D projections of the same output. Actually,

			
θ_1	[3.94335, 3.94335]	[3.71220, 3.71220]	[2.48312, 2.48318]
θ_2	[3.77017, 3.77017]	[3.35355, 3.35356]	[3.95859, 3.95862]
θ_4	[5.51396, 5.51396]	[5.99340, 5.99340]	[2.63872, 2.63877]
θ_5	[3.83643, 3.83643]	[3.97137, 3.97138]	[3.60317, 3.60322]
θ_6	[1.86725, 1.86726]	[2.70201, 2.70202]	[0.68130, 0.68133]
θ_7	[4.69841, 4.69841]	[3.25715, 3.25716]	[5.28944, 5.28951]
θ_8	[2.54508, 2.54508]	[1.46203, 1.46204]	[1.78324, 1.78326]
θ_9	[0.58905, 0.58906]	[4.25173, 4.25174]	[5.00799, 5.00809]
θ_{10}	[5.22246, 5.22246]	[0.66219, 0.66222]	[4.67617, 4.67623]

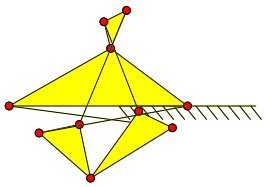
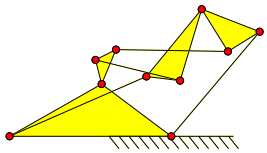
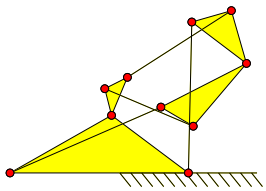
			
θ_1	[2.49296, 2.49301]	[3.03749, 3.03750]	[3.03639, 3.03642]
θ_2	[3.96481, 3.96482]	[1.51266, 1.51266]	[2.19170, 2.19172]
θ_4	[3.02025, 3.02028]	[2.06012, 2.06014]	[2.22075, 2.22080]
θ_5	[3.13912, 3.13917]	[1.19287, 1.19287]	[0.60626, 0.60635]
θ_6	[5.53558, 5.53563]	[3.02443, 3.02445]	[3.27436, 3.27438]
θ_7	[0.97042, 0.97049]	[5.71000, 5.71002]	[3.43216, 3.43218]
θ_8	[2.82075, 2.82078]	[5.74756, 5.74760]	[5.33808, 5.33812]
θ_9	[1.27232, 1.27238]	[1.25375, 1.25376]	[4.26191, 4.26195]
θ_{10}	[3.16983, 3.16988]	[2.82874, 2.82876]	[5.01411, 5.01414]

Table 1: The six configurations of the double butterfly linkage for $\theta_3 = 1.322$ rad (75.75°).

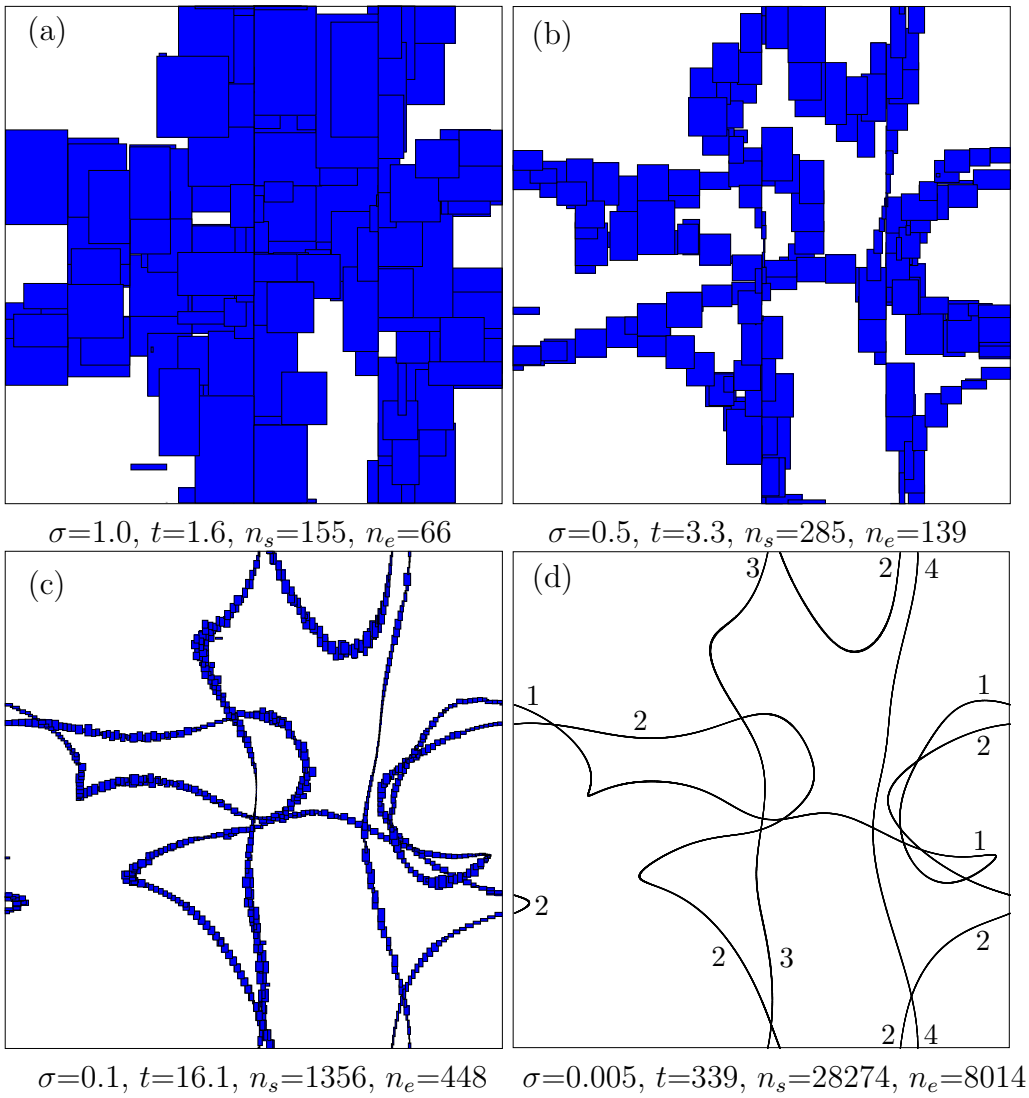


Figure 7: Output boxes at increasing resolution. The horizontal and vertical axes correspond to θ_{10} and θ_5, respectively, spanning the range $[0, 2\pi]$ in all cases. Labels in figure (d) help identifying the four connected components of this linkage configuration space.

four disjoint cyclic paths appear, corresponding to the four possible ways of assembling this mobile linkage. The labels in Fig. 7-(d) help identifying such paths.

To the authors' knowledge, the only box approximations reported so far for the mobile double-butterfly linkage are those in [2], derived with a linear relaxation technique. A significant difference is that [2] employs absolute orientation angles for the links, while the present technique uses the relative angles between them, which allows a natural definition of constraints on the joints' ranges of movement. It is worth noting that, in principle, elimination methods like [15, 16] can also be applied to this mobile linkage. However, these methods are targeted to zero-dimensional solution spaces, and could only generate plots similar to that in Fig. 7-(d) by iteratively sampling an input angle, and solving for the rest of angles at each iteration. Although in general this approach would yield accurate plots for dense-enough samplings, it could be problematic in some cases. For example, on linkages with both rigid and mobile assembly modes, it would fail to detect the former modes, since isolated points in configuration space would almost never get sampled. Difficulties could also arise when the value of the input angle does not determine the overall configuration of the linkage.

8. Conclusions

This work fits within a larger effort aimed at providing a general tool for the position analysis of arbitrary multi-loop linkages, using branch-and-prune algorithms of the kind presented. At the core of such algorithms there always exists a procedure for pruning the solution set of the linkage within a rectangular box of the search space. The "degree of exactness" of this procedure is crucial to attain efficiency in the search process. Note that, ideally, one would like to generate a search tree without empty boxes, and this can only be assured if the pruning procedure always returns the exact bounds. To the authors' knowledge, this is the first time that an exact pruning procedure is given for planar single-loop linkages. On single-loop linkages, thus, the algorithm is advantageous over related algorithms like [2], which would only provide exact bounds in the limit of an iterative process. On multi-loop linkages, as a counterpart, the method in [2] is advantageous in that it prunes the boxes using linear relaxations of all equations simultaneously. Both methods exhibit similar convergence times on the reported experiments, but further tests should be carried out to elucidate how their performance scales comparatively, on linkages of increasing complexity. On the other hand, the method in [2] can be generalized to deal with spatial linkages [3], while the method presented here is generalizable to spherical linkages and a restricted class of

spatial linkages using the results in [8, 19]. The extension of the presented technique to deal with general spatial linkages is a point deserving further attention.

It is worth noting that the *Find Range* algorithm presented in Section 2 is of interest in itself, since it can be used to readily obtain feasible configurations of single-loop linkages by successively fixing one angle after another. The process would start by selecting any variable in the loop, finding its feasibility range, and then taking some value in this range. The loop equation would then be rewritten without the selected variable as explained in Section3, and the process would be applied recursively with the remaining variables, until all of them become fixed. Provided that the link lengths fulfill the required closure condition, reaching a feasible solution is granted independently of the value chosen for each variable. The same procedure would also be applicable to multi-loop linkages, though at the greater cost implied by the iterative nature of the solution process in this case. This approach may be useful in applications that require finding feasible configurations of the linkage, like in probabilistic path planning for closed kinematic chains [20–22], since it allows a directed generation of valid samples in configuration space, avoiding inefficient trial-and-error strategies based on an undirected search in the whole parameter space.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Education and Science through the I+D project DPI2010-18449.

References

[1] J.-P. Merlet, Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis, The International Journal of Robotics Research 23 (3) (2004) 221–235.

[2] J. M. Porta, L. Ros, T. Creemers, F. Thomas, Box approximations of planar linkage configuration spaces, ASME Journal of Mechanical Design 129 (2007) 397.

[3] J. M. Porta, L. Ros, F. Thomas, A linear relaxation technique for the position analysis of multi-loop linkages, IEEE Trans. Robot. 25 (2) (2009) 225–239.

[4] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, Applied interval analysis, Vol. 66, Springer London, 2001.

[5] A. Castellet, Solving inverse kinematics problems using interval methods, Ph.D. thesis, Technical University of Catalonia (1998).

[6] R. S. Rao, A. Asaithambi, S. K. Agrawal, Inverse kinematic solution of robot manipulators using interval analysis, ASME Journal of Mechanical Design 120 (1998) 147–150.

[7] O. Didrit, M. Petitot, E. Walter, Guaranteed solution of direct kinematic problems for general configurations of parallel manipulators, IEEE Trans. on Robotics and Automation 14 (2) (1998) 259–266.

[8] E. Celaya, C. Torras, Solving multiloop linkages with limited-range joints, Mechanism and Machine Theory 29 (3) (1994) 373–391.

[9] R. Moore, R. Kearfott, M. Cloud, Introduction to interval analysis, Society for Industrial Mathematics, 2009.

[10] E. Celaya, C. Torras, On finding the set of inverse kinematic solutions for redundant manipulators, in: J. Angeles, G. Hommel, P. Kovács (Eds.), Computational Kinematics, Kluwer Academic Publishers, 1993, pp. 85–94.

[11] G. Chartrand, L. Lesniak, Graphs and Digraphs, 3rd Edition, Chapman and Hall, 1996.

[12] D. M. Chickering, D. Geiger, D. Heckerman, On finding a cycle basis with a shortest maximal cycle, Information Processing Letters (54) (1994) 55–58.

[13] K. R. Apt, The essence of constraint propagation, Theoretical Computer Science 221 (1-2) (1999) 179–210.

[14] F. Rossi, P. Van Beek, T. Walsh, Handbook of constraint programming, Vol. 35, Elsevier Science, 2006.

[15] J. Nielsen, B. Roth, Solving the input/output problem for planar mechanisms, ASME Journal of Mechanical Design 121 (1999) 206–211.

[16] C. W. Wampler, Solving the kinematics of planar mechanisms by Dixon's determinant and a complex plane formulation, ASME Journal of Mechanical Design 123 (2001) 382–387.

[17] J. Verschelde, Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation, ACM Transactions on Mathematical Software 25 (2) (1999) 251–276.

[18] A. J. Sommese, J. Verschelde, C. W. Wampler, Advances in polynomial continuation for solving problems in kinematics, ASME Journal of Mechanical Design 126 (2004) 262–268.

[19] E. Celaya, Interval propagation for solving parallel spherical mechanisms, in: J. Lenarčič, F. Thomas (Eds.), Advances in Robot Kinematics, 2002, pp. 415–422.

[20] J. Cortés, T. Siméon, Sampling-based motion planning under kinematic loop-closure constraints, in: Workshop on Algorithmic Foundations of Robotics, Springer, 2005, pp. 75–90.

[21] M. Stilman, Task constrained motion planning in robot joint space, in: IEEE International Conference on Intelligent Robots and Systems, IEEE, 2007, pp. 3074–3081.

[22] D. Berenson, S. Srinivasa, D. Ferguson, J. Kuffner, Manipulation planning on constraint manifolds, in: IEEE International Conference on Robotics and Automation, IEEE, 2009, pp. 625–632.