Solution of the Direct Geometrico-Static Problem of 3-3 Cable-Driven Parallel Robots by Interval Analysis: Preliminary Results

Alessandro Berti, Jean-Pierre Merlet and Marco Carricato

Abstract This paper addresses the direct geometrico-static analysis of under-constrained cable-driven parallel robots with 3 cables. The task consists in finding all equilibrium configurations of the end-effector when the cable lengths are assigned. An interval-analysis-based procedure is proposed to numerically find the real solutions of the problem for a robot of generic geometry. Three equation sets obtained by different approaches are implemented in the problem-solving algorithm and a comparison between the main advantages and disadvantages of each one of them is reported.

1 Introduction

Cable-driven parallel robots (CDPRs) employ cables in place of rigid-body extensible legs in order to control the end-effector posture. CDPRs strengthen classic advantages characterizing closed-chain architectures versus serial ones, like reduced mass and inertias, a larger payload to robot weight ratio, high dynamic performances, improved motor efficiency, etc., while providing peculiar advantages, such as a larger workspace, reduced manufacturing and maintenance costs, ease of assembly and disassembly, high transportability, and superior modularity and reconfigurability. Over the past decades, these characteristics increased researchers' interest for this kind of manipulators.

A CDPR is *fully-constrained* if the pose of the end-effector is completely determined when actuators are locked and, thus, all cable lengths are assigned. A CDPR is, instead, *under-constrained* if the end-effector preserves some degrees of

Alessandro Berti and Marco Carricato: Dept. of Mechanical Engineering (DIEM), University of Bologna, Bologna, Italy. E-mail: alessandro.berti10@unibo.it, marco.carricato@unibo.it.

Jean-Pierre Merlet: COPRIN Project, French National Institute for Research in Computer Science and Control (INRIA), Sophia-Antipolis, France. E-mail: jean-pierre.merlet@sophia.inria.fr.

freedom (dofs) once actuators are locked [16, 2]. This occurs either when the end-effector is controlled by a number of cables n smaller than the number of dofs that it possesses with respect to the base or when some cables become slack in a fully-constrained robot. The use of CDPRs with a limited number of cables is justified in several applications (such as, for instance, rescue, service or rehabilitation operations [19, 18, 14]), in which the task to be performed requires a limited number of controlled freedoms (only n dofs may be governed by n cables) or a limitation of dexterity is acceptable in order to decrease complexity, cost, set-up time, likelihood of cable interference, etc. Furthermore, a theoretically fully-constrained CDPR may operate, in appreciable parts of its geometric workspace, as an under-constrained robot, namely when a full restraint of the end-effector may not be achieved because it would require a negative tension in one or more cables. Even though the above considerations motivate a careful study of under-constrained CDPRs, little research was dedicated to them [21, 6, 7, 15, 10, 2, 3, 4, 1, 5].

A major challenge in the kinetostatic analysis of under-constrained CDPRs comes from the fact that, when the actuators are locked and the cable lengths are assigned, the end-effector is still movable, so that the actual configuration is determined by the applied forces. Accordingly, *loop-closure* and *mechanical-equilibrium equations* must be simultaneously solved and displacement analyses, which are aimed at determining the overall robot configuration when a set of n variables is assigned, become *geometrico-static problems* [2]. These are considerably more complicated than the displacement analyses of rigid-link parallel manipulators [11] and only limited results were presented so far [6, 15, 10].

Only recently Carricato and Merlet [2] proposed a general methodology for the kinematic, static and stability analysis of general under-constrained n-n CDPRs, i.e. manipulators in which a fixed base and a mobile platform are connected to each other by n cables, with $n \leq 5$ and with cable exit points on the base and anchor points on the platform being distinct. A successful implementation of this methodology, based on exact-arithmetic elimination procedures, allowed the direct geometrico-static problem (DGP) of the 3-3 CDPR to be solved [3]. In particular, a least-degree univariate polynomial in the ideal generated by the equations governing the robot model was found and the DGP of the 3-3 CDPR was proven to admit at the most 156 solutions in the complex field. However, the approach used in [3] has the following drawbacks.

- Elimination by exact-arithmetic-based procedures requires equations with rational coefficients. However, when geometrical parameters are approximated by rationals having large integer denominators and numerators, the size of the coefficients of the resulting polynomials may be extremely large and very difficult to manage. In addition, solving high-order polynomials in a reliable way may be difficult, as the calculation of coefficients is very sensitive to numerical errors.
- It is not possible to incorporate constraints on the unknowns, so that all roots (both complex and real, regardless of the tension sign) must be calculated and then they need to be post-processed in order to discard unfeasible ones.

Effective alternatives are represented by approaches based on floating-point arithmetic, such as homotopy continuation or interval analysis. In this paper, a method based on interval analysis is proposed. This computing technique was shown to be very efficient in solving the direct kinematics of rigid-link parallel robots [12], but its efficiency is strictly related to the heuristics incorporated in the problem-solving algorithm. Indeed, the computation time for a given problem may vary from a few seconds, if the right heuristics are adopted, to several hours with a poor implementation.

This paper is organized as follows. Section 2 provides basilar notions of interval analysis. Section 3 presents the geometrico-static model of the 3-3 CDPR: the set of variables representing the system configuration is described, and three different methods to formulate the equations governing the DGP are discussed. Section 4 describes the structure of the code and the procedures incorporated therein, whereas in Section 5 the results obtained from some case studies are presented. Section 6 draws some conclusions.

2 Interval analysis

A short introduction to interval analysis is presented in the following. More informations may be found in [9, 17].

The *real interval* $X = [\underline{x}, \overline{x}]$ is defined as the set of real numbers y such that $\underline{x} \leq y \leq \overline{x}$. The *width* of the interval is $\overline{x} - \underline{x}$ and its *mid-point* is $(\overline{x} + \underline{x})/2$. An *interval vector* $\mathbf{X}$, also called a *box*, is a list of intervals. The mid-point of a box is the vector whose components are the mid-points of its interval components.

If $f(\mathbf{x})$ is a function in n unknowns, with $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, and $\mathbf{B} = [X_1, X_2, \ldots, X_n]$ is a box comprising an interval for each unknown, an interval evaluation $F(\mathbf{B})$ of f over $\mathbf{B}$ is an interval $\left[\underline{F}, \overline{F}\right]$ such that, for any $\mathbf{x} \in \mathbf{B}$, $\underline{F} \leq f(\mathbf{x}) \leq \overline{F}$. There are many ways to implement an interval evaluation of a function but the simplest one is the *natural evaluation*, in which each mathematical operator is substituted by an interval equivalent. For example, if $f(x) = x^2 - 2x + 1$ and $X = [4,5]$, the natural evaluation of f over X is:

$$f([4,5]) = [4,5]^2 - 2[4,5] + 1 = [16,25] - [8,10] + [1,1] = [7,18] \qquad (1)$$

It is worth emphasizing that the bounds provided by the natural evaluation of f are not exact: the upper (lower) bound may be larger (lower) than the actual maximum (minimum) of the function image, namely $f(\mathbf{B}) = \{f(\mathbf{x}) | x \in \mathbf{B}\} \subseteq F(\mathbf{B})$. The overestimation ordinarily decreases with the width of the box over which f is evaluated, and there are cases and methods that allow one to get bounds as tight as possible.

The following properties hold:

- if $0 \notin \left[\underline{F}, \overline{F}\right]$, then there is no value of $\mathbf{x}$ such that $f(\mathbf{x}) = 0$ (Property A);

- the bounds of F are exactly the minimum and the maximum of $f(\mathbf{B})$ when f may be expressed so as to contain a single occurrence of each unknown x_i $(i = 1,\ldots,n)$ (Property B);
- interval evaluation may be implemented on a computer in a 'guaranteed' way, by taking into account numerical round-off errors;
- interval arithmetic is not restricted to algebraic functions, but it may be used for all mathematical functions of engineering relevance.

The structure of a generic interval-analysis-based algorithm to solve a system of n equations in n unknowns is as follows. Let $\mathbf{B}_1 = [X_1, X_2, \ldots, X_n]$ be a box and $\mathbf{f} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})]$ a vector equation to be solved within $\mathbf{B}_1$. $\mathcal{L}$ is a list of boxes, initially set as $\mathcal{L} = \{\mathbf{B}_1\}$. An index i, initialized to 1, indicates which box $\mathbf{B}_i$ in $\mathcal{L}$ is currently being processed, while N denotes the number of boxes in the list. The interval evaluation of f_j over $\mathbf{B}_i$ is denoted as $F_j(\mathbf{B}_i)$, with $j = 1,\ldots,n$. A key element in the algorithm is the evaluation operator $\mathcal{E}$, which takes a box $\mathbf{B}_i$ as input and it returns:

- 1 if the width of $F_j(\mathbf{B}_i)$ is lower than a given threshold ε and includes 0 for any j; in this case, $\mathbf{B}_i$ is a solution of the system and it is stored in the solution list $\mathcal{S}$;
- -1 if $F_j(\mathbf{B}_i)$ does not include 0 for at least one j;
- 0 otherwise.

Another fundamental element is the filter operator $\mathcal{F}$, which takes a box as input and it returns:

- -1 if there is no solution in the box;
- a box whose width is smaller than or equal to the width of the input box, after determining that the removed part of the input box cannot contain a solution.

The overall algorithm proceeds along the following steps.

1: $i = 1$, $\mathcal{L} = \{\mathbf{B}_1\}$, $\mathcal{S} = \{\}$, $N = 1$;
2: if $i > N$, then return $\mathcal{S}$;
3: if $\mathcal{F}(\mathbf{B}_i) = -1$, then $i = i+1$, go to 2, else $\mathbf{B}_i = \mathcal{F}(\mathbf{B}_i)$;
4: compute $\mathcal{E}(\mathbf{B}_i)$
   a) if $\mathcal{E}(\mathbf{B}_i) = -1$, then $i = i+1$, go to 2;
   b) if $\mathcal{E}(\mathbf{B}_i) = 1$, then add $\mathbf{B}_i$ to $\mathcal{S}$, $i = i+1$, go to 2;
   c) if $\mathcal{E}(\mathbf{B}_i) = 0$, select an unknown x_k and bisect X_k in the middle point, create two new boxes $\mathbf{B}_i'$ and $\mathbf{B}_i''$ from $\mathbf{B}_i$ and add them to $\mathcal{L}$, i.e. $\mathcal{L} = \mathcal{L} \cup \{\mathbf{B}_i', \mathbf{B}_i''\}$, $N = N+2$, $i = i+1$, go to 2.

The above algorithm always terminates, since the size of a box always decreases after a bisection. Provided that the new boxes resulting from a bisection are put at the top of the list, there is usually no problem of memory storage.

The efficiency of the described algorithm mainly depends on the effectiveness of the operators $\mathcal{E}$ and $\mathcal{F}$, and thus on the heuristics adopted to implement them. In Section 4, some important tools of interval analysis are presented, which drastically reduce the computation time.

3 Geometrico-static model

A general under-constrained 3-3 CDPR comprises a mobile platform connected to a fixed base by 3 cables. The ith cable $(i = 1,2,3)$ exits from the base at point A_i and it is connected to the platform at point B_i (Fig. 1). The platform is acted upon by a force of constant magnitude Q applied at point G, e.g. the platform weight acting through its center of mass. This force is described as a 0-pitch wrench $Q\mathcal{L}_e$, where $\mathcal{L}_e$ is the normalized Plücker vector of its line of action.

$Oxyz$ is a Cartesian coordinate frame attached to the base, with origin in O, whereas $O'x'y'z'$ is a Cartesian frame appended to the moving platform, with origin in O'. Without loss of generality, the coordinate frames are chosen in such a way that $O \equiv A_1$, $O' \equiv B_1$, the z axis is directed as $\mathcal{L}_e$, point A_2 lies in plane xz and point B_2 lies in plane $x'z'$. By this choice, the position vectors of points A_1, A_2, A_3 and B_1, B_2, B_3, G in frame $Oxyz$ and of points B_1, B_2, B_3 and G in frame $O'x'y'z'$ may be expressed as

$$\begin{aligned} \mathbf{a}_1 &= \left[0,0,0\right]^{\mathrm{T}} \qquad \mathbf{a}_2 = \left[a_{21},0,a_{23}\right]^{\mathrm{T}} \qquad \mathbf{a}_3 = \left[a_{31},a_{32},a_{33}\right]^{\mathrm{T}}, \\ \mathbf{b}_1 &= \left[x_1,y_1,z_1\right]^{\mathrm{T}}\ \mathbf{b}_2 = \left[x_2,y_2,z_2\right]^{\mathrm{T}}\ \mathbf{b}_3 = \left[x_3,y_3,z_3\right]^{\mathrm{T}}\ \mathbf{g} = \left[g_1,g_2,g_3\right]^{\mathrm{T}}, \\ \mathbf{b}'_1 &= \left[0,0,0\right]^{\mathrm{T}}\ \mathbf{b}'_2 = \left[b'_{21},0,b'_{23}\right]^{\mathrm{T}}\ \mathbf{b}'_3 = \left[b'_{31},b'_{32},b'_{33}\right]^{\mathrm{T}}\ \mathbf{g}' = \left[g'_1,g'_2,g'_3\right]^{\mathrm{T}}, \end{aligned} \tag{2}$$

with a_{ij}, b'_{ij} and g'_j $(i,j = 1,2,3)$ being known geometric parameters and x_i, y_i and z_i $(i = 1,2,3)$ being the variables describing the posture of the platform.

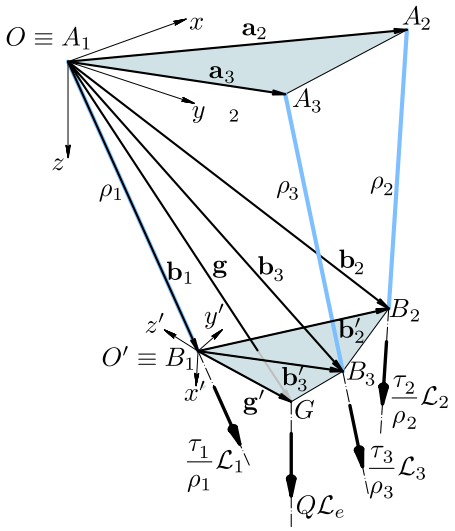

Fig. 1 Geometric model of a cable driven parallel robot with three cables.

3.1 Fundamental geometric and static equations

If ρ_i is the assigned length of the ith cable (taken as strictly positive), when all cables are *active* (i.e. in tension) the set of geometrical constraints imposed on the platform comprises 3 relations, namely

$$\|\mathbf{b}_i - \mathbf{a}_i\|^2 = \rho_i^2, \quad i = 1,2,3. \tag{3}$$

Since the platform has 6 dofs, its posture is ultimately determined by mechanical equilibrium. The normalized Plücker vector of the line associated with the ith cable is $\mathcal{L}_i/\rho_i$, where, in axis coordinates, $\mathcal{L}_i = \left[(\mathbf{a}_i - \mathbf{b}_i)\,;\, \mathbf{p}_i \times (\mathbf{a}_i - \mathbf{b}_i)\right]$ where $\mathbf{p}_i$ is any vector from an arbitrarily-chosen reference point P (called, for brevity, *moment pole*) to the cable line. Accordingly, the wrench exerted by the ith cable on the platform is $(\tau_i/\rho_i)\,\mathcal{L}_i$, with τ_i being a positive scalar representing the intensity of the cable tensile force. Static equilibrium may then be expressed as

$$\underbrace{\left[\mathcal{L}_1 \;\; \mathcal{L}_2 \;\; \mathcal{L}_3 \;\; \mathcal{L}_e\right]}_{\mathbf{M}(P)} \begin{bmatrix} \tau_1/\rho_1 \\ \tau_2/\rho_2 \\ \tau_3/\rho_3 \\ Q \end{bmatrix} = \mathbf{0}\,, \tag{4}$$

with $\tau_i \geq 0$, $i = 1,2,3$.

When the direct geometrico-static problem (DGP) is solved, the cable lengths are assigned. Accordingly, Eqs. (3) and (4) form a coupled system of 9 equations whose unknowns are the platform-pose variables, grouped in the array $\mathbf{X}$, and the cable tensions, grouped in the array $\boldsymbol{\tau}$. The efficiency of the interval-analysis-based problem-solving algorithm is strictly related to the complexity of the involved equations. In particular, for each equation, the occurrences of the same variable should be limited as much as possible (Property B). In this perspective, the choice of the most suitable *parameterization of the platform pose* and *formulation of the static constraints* is extremely important. These issues will be discussed in the following sections.

3.2 Parameterization of the platform pose

The platform pose $\mathbf{X}$ is described by 9 variables, namely the components of the position vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ in the $Oxyz$ frame (Fig. 1), which must satisfy the following geometrical constraints:

$$\begin{aligned} (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 - \|\mathbf{b}_2'\|^2 &= 0 \\ (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 - \|\mathbf{b}_3'\|^2 &= 0 \\ (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2 - \|\mathbf{b}_3' - \mathbf{b}_2'\|^2 &= 0\,. \end{aligned} \tag{5}$$

By this parameterization, the relationships in Eq. (3) assume the form

$$\begin{aligned} x_1^2+y_1^2+z_1^2-\rho_1^2=0 \\ (x_2-a_{21})^2+y_2^2+(z_2-a_{23})^2-\rho_2^2=0 \\ (x_3-a_{31})^2+(y_3-a_{32})^2+(z_3-a_{33})^2-\rho_3^2=0 \end{aligned} \tag{6}$$

and the coordinates of point G in $Oxyz$ may be expressed as

$$\mathbf{g}=\mathbf{b}_1+\alpha\,(\mathbf{b}_2-\mathbf{b}_1)+\beta\,(\mathbf{b}_3-\mathbf{b}_1)+\gamma[(\mathbf{b}_2-\mathbf{b}_1)\times(\mathbf{b}_3-\mathbf{b}_1)]\ , \tag{7}$$

where α, β and γ are known constants obtained by solving the system

$$\alpha\,\mathbf{b}_2'+\beta\,\mathbf{b}_3'+\gamma\left(\mathbf{b}_2'\times\mathbf{b}_3'\right)-\mathbf{g}'=\mathbf{0}\,. \tag{8}$$

Accordingly, by choosing O as the moment pole, matrix $\mathbf{M}(P)$ in Eq. (4) may be explicitly written as

$$\mathbf{M}(O)=\begin{bmatrix} x_1 & x_2-a_{21} & x_3-a_{31} & 0 \\ y_1 & y_2 & y_3-a_{32} & 0 \\ z_1 & z_2-a_{23} & z_3-a_{33} & -1 \\ 0 & -a_{23}y_2 & a_{32}z_3-a_{33}y_3 & -y_g \\ 0 & a_{23}x_2-a_{21}z_2 & a_{33}x_3-a_{31}z_3 & x_g \\ 0 & a_{21}y_2 & a_{31}y_3-a_{32}x_3 & 0 \end{bmatrix} \tag{9}$$

and static equations become

$$x_1\frac{\tau_1}{\rho_1}+(x_2-a_{21})\frac{\tau_2}{\rho_2}+(x_3-a_{31})\frac{\tau_3}{\rho_3}=0 \tag{10a}$$

$$y_1\frac{\tau_1}{\rho_1}+y_2\frac{\tau_2}{\rho_2}+(y_3-a_{32})\frac{\tau_3}{\rho_3}=0 \tag{10b}$$

$$z_1\frac{\tau_1}{\rho_1}+(z_2-a_{23})\frac{\tau_2}{\rho_2}+(z_3-a_{33})\frac{\tau_3}{\rho_3}-Q=0 \tag{10c}$$

$$-a_{23}y_2\frac{\tau_2}{\rho_2}+(a_{32}z_3-a_{33}y_3)\frac{\tau_3}{\rho_3}-Qy_g=0 \tag{10d}$$

$$(a_{23}x_2-a_{21}z_2)\frac{\tau_2}{\rho_2}+(a_{33}x_3-a_{31}z_3)\frac{\tau_3}{\rho_3}+Qx_g=0 \tag{10e}$$

$$a_{21}y_2\frac{\tau_2}{\rho_2}+(a_{31}y_3-a_{32}x_3)\frac{\tau_3}{\rho_3}=0 \tag{10f}$$

Equations (6), (5) and (10) form a square system of 12 scalar relations in the 12 variables grouped in

$$\mathbf{Y}=\left[\mathbf{X}^{\mathrm{T}},\boldsymbol{\tau}^{\mathrm{T}}\right]^{\mathrm{T}}=\left[x_1,y_1,z_1,x_2,y_2,z_2,x_3,y_3,z_3,\tau_1,\tau_2,\tau_3\right]^{\mathrm{T}}. \tag{11}$$

The polynomial relations in Eqs. (5) and (6) have degree 2 in $\mathbf{X}$, whereas the relations in Eq. (10) have degree 2 in $\mathbf{X}$, degree 1 in $\boldsymbol{\tau}$ and degree 2 in $\mathbf{Y}$.

It is worth observing that the described parameterization, which uses 9 variables, is redundant, since a minimal representation of the platform pose may be achieved by means of only 6 (cf. [3]). However, the described parameterization is preferred here, since it yields simpler lower-order polynomials, which prove to be stabler and more efficient when interval analysis methods are implemented to solve them. In fact, by property B (Section 2), using simpler expressions is valuable even at the price of introducing a larger number of variables, in order to limit overestimation in interval evaluation.

3.3 Formulation of the static constraints

The direct geometrico-static problem (DGP) requires simultaneously solving the relations emerging from both the geometrical and the static constraints. According to Section 3.2, these constraints may be set up as a system of 12 equations having $\mathbf{Y}$ as unknown, namely Eqs. (5), (6) and (10). These equations are implemented in the first solution routine (called R1).

The number of unknowns may be decreased by eliminating cable tensions. In fact, by observing that Eq. (10) is linear in τ_1, τ_2 and τ_3, 3 linearly independent relationships may be selected within the system, say (10a), (10d) and (10e), and solved for the tensions as unknowns. The tensions this way calculated may be substituted back into Eqs. (10b), (10c), (10f), thus forming a system of 3 equations in $\mathbf{X}$ only, namely Eqs. (11b$'$), (11c$'$) and (11f$'$). The system of equations implemented in the second solution routine (R2) comprises these relationships, together with Eqs. (5) and (6). The resulting system comprises 9 equations in 9 unknowns (i.e. $\mathbf{X}$).

An alternative, more elaborated strategy to eliminate cable tensions, presented in [3], may be designed by observing that Eq. (4) admits a solution only if

$$\operatorname{rank}\left[\mathbf{M}(P)\right] \leq 3\,. \tag{12}$$

Accordingly, by setting *all* 4×4 minors of $\mathbf{M}(P)$ equal to zero and by conveniently changing the moment pole, a large set of linearly independent relations only comprising the platform-pose variables may be derived, i.e.

$$p_k\left(\mathbf{X}\right) = 0, \quad k = 1\ldots N_p, \tag{13}$$

where N_p is an integer significantly larger than the number N_X of variables contained in $\mathbf{X}$. For the DGP to admit a solution, the above equations must be dependent, though in a non-linear way. When complemented with Eqs. (5) and (6), Eq. (13) allows the pose $\mathbf{X}$ to be directly solved. The price paid for the elimination of cable tensions is that the polynomials comprised in Eq. (13) are much more involved than those in Eq. (10). In particular, they have a higher degree, more terms and more complicated coefficients. A partial simplification is obtained as follows [3].

Since the moment vector of the first column of $\mathbf{M}(O)$ is zero (Eq. (9)), setting $\det \mathbf{M}_{j456,1234}(O) = 0$, for $j = 1 \ldots 3$, yields

$$\det \mathbf{M}_{456,234}(O) \left[x_1,\, y_1,\, z_1\right]^{\mathrm{T}} = (\mathbf{b}_1 - \mathbf{a}_1) \det \mathbf{M}_{456,234}(O) = \mathbf{0}\,. \tag{14}$$

Since $(\mathbf{b}_1 - \mathbf{a}_1)$ may not vanish by assumptions ($\rho_1 \neq 0$), Eq. (14) provides

$$\det \mathbf{M}_{456,234}(O) = 0\,. \tag{15}$$

Two analogous equations may be obtained by conveniently changing the moment pole, namely

$$\det \mathbf{M}_{456,134}(A_2) = 0 \tag{16}$$

$$\det \mathbf{M}_{456,124}(A_3) = 0 \tag{17}$$

The system of equations implemented in the third solution routine (R3) is formed by Eqs. (5), (6), (15), (16) and (17).

Interval-analysis methods require each variable to be comprised between a lower and an upper bound. In the equation set implemented in R1, in which cable tensions appear as unknowns, lower and upper bounds for τ_1, τ_2 and τ_3 may be conveniently specified. The lower bound may be straightforwardly set to 0, to avoid solutions with negative cable tension. The upper bound may be chosen, instead, on the basis of the maximum tensile strength of the cables.

Conversely, in the equation sets implemented in R2 and R3, cable tensions may not be bounded, since they do not appear as unknowns, and positive-tension solutions may not be directly looked for. Accordingly, these routines find all solutions contained in a purely geometrical starting box and they isolate those with positive tension in all cables by adding a suitable test to the filter operator $\mathcal{F}$.

4 The problem-solving algorithm

The problem-solving code was developed by using the C++ library ALIAS [13], which contains interval-analysis-based algorithms developed by the INRIA team COPRIN.

4.1 Code structure

The structure outlined hereafter is common to all routines R1, R2 and R3 and it follows the scheme presented in Sec. 2.

The main procedure initially retrieves the geometric data of the manipulator, the initial box defining the search domain and the configuration parameters of the ALIAS functions incorporated into the code, from convenient text files.

The operations performed by the algorithm may be resumed as follows. At the generic ith step, a first filter $\mathcal{F}_1$, which implements the 2B method (described in Sec. 4.4.1), tries to shrink, or even eliminate, the current box $\mathbf{B}_i$. After that, the evaluation operator $\mathcal{E}$ tests if $\mathbf{B}_i$ may contain solutions or not. If the test is negative, the box is discarded. If the test is positive and the width of the box is smaller than a given threshold ε, $\mathbf{B}_i$ is a solution and it is added to the solution list $\mathcal{S}$. If the test is positive, but the width of the box is larger than ε, another filter $\mathcal{F}_2$, which implements the 3B method (described in Sec. 4.4.2), is applied to further contract the box and $\mathbf{B}_i$ is finally bisected. The adopted bisecting strategy consists in splitting the variable having the largest width.

4.2 Domain initialization

The first step of the code is initializing the search domain. For the application to the DGP of a CDPR, these bounds are quite useful, as they allow one to take the geometrical constraints of the robot into account.

The starting intervals for the geometrical unknowns in $\mathbf{X}$ may be easily determined by observing that $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$ have to lie inside the spheres centered, respectively, in A_1, A_2 and A_3 and having radii ρ_1, ρ_2 and ρ_3. For routine R1, initial bounds for cable tensions may be established as explained in Sec. 3.3.

4.3 Evaluation operator

The evaluation operator $\mathcal{E}$ is implemented by means of the ALIAS procedure `Solve_General_Gradient_Interval` (`SGGI`). If the Jacobian matrix of the system to be solved exists and it may be computed, `SGGI` improves interval evaluation of functions by conveniently using gradients and by taking advantage of possible monotonicities [9]. `SGGI` also uses Moore theorem [17] to determine if a unique solution exists in a given box, in which case Krawczyk method is applied to compute the solution [13]. In addition, the *inflation method* [13] is used to increase the width of the box in which the computed solution remains unique, thus working as a filter for the neighboring boxes.

4.4 Filtering operators

The performances of the algorithm largely depend on the filter operators $\mathcal{F}_1$ and $\mathcal{F}_2$. Their basic functioning scheme is described hereafter.

4.4.1 Filter $\mathcal{F}_1$: the 2B method

The 2B filter consists in rewriting each equation as the equality of two terms, determining if the interval evaluations of both terms are consistent and, if not, using consistency to improve the width of the interval for one or more unknowns.

Let, for instance, the equation $x^2 - 2x + 1 = 0$ be considered. By introducing the new variable $X = x^2$, the original equation may be re-written as $X = 2x - 1$. Now, let $[\underline{u}, \overline{u}]$ be the interval evaluation of $2x - 1$. If $\overline{u} > 0$, then the inverse function of X indicates that x should lie in $\left[-\sqrt{\overline{u}}, \sqrt{\overline{u}}\right]$ and, by this information, the current interval of x may be updated. If $\underline{u} > 0$, then the inverse function of X indicates that x should lie outside $\left[-\sqrt{\underline{u}}, \sqrt{\underline{u}}\right]$: if the range of x is included in this interval, then there is no solution to the equation in the current box.

This process may be repeated for each unknown in the equation and for a number of runs depending on the rate of contraction obtained for each interval.

4.4.2 Filter $\mathcal{F}_2$: the 3B method

By this approach, the range $\left[\underline{x}_j, \overline{x}_j\right]$ for one variable x_j in a given box **B** is replaced by $\left[\underline{x}_j, \underline{x}_j + \delta\right]$, where δ is an arbitrary small number, while the ranges for the other variables remain unchanged. Then, the algorithm tests whether, for the new ranges, the system may have some solution, either by using the 2B method and/or by evaluating the equations. If the answer is negative, the range for x_j in the box **B** is changed to $\left[\underline{x}_j + \delta, \overline{x}_j\right]$. The process is then repeated on the new range, but the width of the test interval is now doubled, i.e. the algorithm tests the interval $\left[\underline{x}_j, \underline{x}_j + 2\delta\right]$. The process is repeated until the no-solution test is no longer satisfied. Within the 3B filter, the 2B method may also be applied to update the range for all unknowns.

The same process may be repeated on the right side of the interval, by trying to decrease the upper bound of the range for x_j (in this case, the interval test is $[\overline{x}_j - \delta, \overline{x}_j]$).

4.5 Parallel implementation

Most interval-analysis-based algorithms are appropriate for a distributed implementation. Indeed, processing a given box does not generally depend on the processing of the other boxes in the list. The implementation may be as follows. A master com-

puter manages the list and it sends a box to a slave computer. The slave executes the algorithm, by performing a few bisections. Then, it returns the remaining boxes to the master and it is ready to process another box. Such a scheme may be easily implemented in a network of workstations. The decrease of computation time will be, in general, less than proportional to the number of slaves, due to the overhead of the data transmission between the master and the slaves.

This approach may also take advantage of modern multi-core CPU architectures. By following this scheme and by using POSIX thread libraries[1], a distributed implementation of the DGP code was prepared and used on a single workstation with a multi-core CPU. In the first step, an instance of $\mathcal{E}$ generates a few boxes and store them in the list $\mathcal{L}$. Then, a number of threads equal to the number of CPUs is created, with each one taking a box from $\mathcal{L}$. A "local" instance of $\mathcal{E}$ performs an assigned number of bisections and it appends the generated boxes to $\mathcal{L}$. The solutions found, if any, are appended to the solution list $\mathcal{S}$. Even though implementing this algorithm on a single machine is not as effective as a distributed implementation over a computer network, the results are quite good. In the following, the routines incorporating the parallel-computing scheme are denoted by an asterisk (*).

4.6 Routine configuration

The main configuration parameters of each routine are reported hereafter.

R1:
- The equations to be solved are Eqs. (5), (6) and (10a)–(10f).
- The 2B filter is applied to Eqs. (5), (6), (10a), (10b), (10c) and (10f).
- The 3B filter is applied to all the equations of the system.
- Positive-tension configurations are found by suitably configuring the initial search domain.

R2:
- The equations to be solved are Eqs. (5), (6) and Eqs. (11b′), (11c′) and (11f′).
- The 2B filter is applied to Eqs. (5), (6) and (11b′). Applying the 2B filter to the other equations is not convenient in terms of computation time.
- The 3B filter is applied only to Eqs. (5) and (6), since Eqs. (11b′), (11c′) and (11f′) are too complex and they would excessively raise the computational burden.
- Positive-tension configurations are obtained by introducing, in filter $\mathcal{F}_1$, a simplification procedure that calculates tensions from Eqs. (10a), (10d) and (10e) and discards the boxes in which positive tensions do not appear.

R3:
- The equations to be solved are Eqs. (5), (6) and (15)–(17).
- The 2B filter is applied to Eqs. (5), (6) and (15).
- The 3B filter is applied to Eqs. (5) and (6).
- Positive-tension configurations are obtained as in R2.

[1] More informations may be found on the Linux manual.

5 Discussion of results

5.1 Performances and possible improvements

Extensive numerical investigation was performed to test the efficiency and robustness of the code, as well as to show the performances of the different routines that were implemented. Two meaningful examples are reported in section 5.2.

R1 is the only routine that offers the possibility to specify cable-tension bounds in the initial search domain. It is particularly stable, even though not particularly fast. R2 is usually faster than R1, mainly because of the reduced number of unknowns incorporated in the static-equation formalization. However, its system of equation may often become nearly singular and this produces very high computation times, making this routine almost useless. Choosing different sets of relationships in Eqs. (10) to calculate cable tensions does not seem to improve the routine performances. R3 is stable and reliable. For ordinary robot geometries, in which the base has larger dimensions than the platform and cable lengths are supposed to position the platform well within the workspace, R3 is usually faster than R1 while when the base and the platform have similar dimensions R1 is more effective than R3.

A number of possible improvements may be conceived in order to enhance the efficiency of the code. In particular, the evaluation operator $\mathcal{E}$ may be improved by using Kantorovitch theorem [20] instead of Moore's one, to verify if a single solution exists in a given box. Kantorovitch theorem should speed up computation, especially for simple equations such as those implemented in R1. Furthermore, the 2B filter $\mathcal{F}_1$ may be enhanced by introducing additional tests based on larger sets of relations chosen from the minors of matrix $\mathbf{M}$ (cf. Eq. (9)). Indeed, when dealing with interval analysis, additional equations may allow lower computation times to be attained, as they enrich the set of available tests that may be used to exclude portions of the domain from the solution search. Another possible improvement may be obtained by using Rohn extremal test [8] in order to check if $\mathbf{M}$ is rank-deficient or not. This variants will be implemented in an upgraded version of the code. This will also be able to filter stable equilibrium configurations among the admissible ones (cf. [2]).

5.2 Examples

The results reported in the following two examples were obtained by a personal computer Toshiba® with processor Intel® Core i7 CPU M620, 2.67 GHz, equipped with 2 cores and 4 threads. All lengths are expressed in meters and the load applied on the platform is in newtons. For both examples, the threshold ε defined in Section 4.1 is set equal to 0.001, whereas the parameter δ of the 3B filter discussed in Section 4.4.2 is set equal to 0.01. The geometric parameters, the search domains and the solutions therein found are reported in Tables 1 and 2, whereas Tables 3 and

4 list, for the two examples, the computation times required by the three routines involved. In Example 1, the base is considerably larger than the platform, as it is likely to occur in practice, whereas in Example 2 the dimensions of the two links are almost the same, so that (at the equilibrium) the cables are almost parallel. Although the latter example has little practical interest, it emphasizes the fact that the system of equation adopted in R2 becomes almost singular and thus the computation time increases exponentially.

Table 1 Geometric parameters, search domain and solutions for Example 1.

Data	$\mathbf{a}_2=[10,0,0]^{\mathrm{T}}$ $\mathbf{a}_3=[0,12,0]^{\mathrm{T}}$ $(\rho_1,\rho_2,\rho_3)=[7.5,10.0,9.5]$ $Q=1$ $\mathbf{b}'_2=[1.41,0,0]^{\mathrm{T}}$ $\mathbf{b}'_3=[0.71,0.71,1]^{\mathrm{T}}$ $\mathbf{g}'=[0.71,0.71,0]^{\mathrm{T}}$											
Search Domain	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	τ_1	τ_2	τ_3
	−7.5	−7.5	−7.5	0	−10	−10	−9.5	2.5	−9.5	0	0	0
	7.5	7.5	7.5	20	10	10	9.5	21.5	9.5	2	2	2
	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	τ_1	τ_2	τ_3
Results	1.682	3.674	6.318	2.800	4.126	5.580	1.933	5.060	6.193	0.546	0.326	0.550
	1.379	4.481	5.854	2.623	3.904	5.507	1.535	3.834	4.606	0.684	0.305	0.614
	2.787	4.995	4.851	3.511	6.151	4.478	4.200	4.959	4.800	0.289	0.787	0.912
	3.515	4.063	5.233	2.475	3.244	5.732	2.260	4.630	5.553	0.526	0.511	0.581
	3.603	5.329	3.857	2.233	4.976	3.863	3.224	3.968	3.916	0.590	0.783	0.956
	1.344	3.471	6.511	2.397	3.329	5.578	1.359	4.259	5.337	0.676	0.251	0.486

Table 2 Geometric parameters, search domain and solutions for Example 2.

Data	$\mathbf{a}_2=[10,0,0]^{\mathrm{T}}$ $\mathbf{a}_3=[0,12,0]^{\mathrm{T}}$ $(\rho_1,\rho_2,\rho_3)=[7.5,10.0,9.5]$ $Q=1$ $\mathbf{b}'_2=[9.90,0,0]^{\mathrm{T}}$ $\mathbf{b}'_3=[4.98,4.98,7]^{\mathrm{T}}$ $\mathbf{g}'=[4.98,4.98,0]^{\mathrm{T}}$											
Search Domain	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	τ_1	τ_2	τ_3
	−7.5	−7.5	−7.5	0	−10	−10	−9.5	2.5	−9.5	0	0	0
	7.5	7.5	7.5	20	10	10	9.5	21.5	9.5	2	2	2
	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3	τ_1	τ_2	τ_3
Results	5.537	4.500	0.771	3.925	−1.862	7.722	−1.787	6.214	7.320	0.682	0.660	0.545
	3.131	−1.218	6.705	1.220	4.679	−1.013	4.909	8.500	7.341	0.766	0.655	0.494
	−1.423	1.949	7.101	7.949	4.796	8.532	4.475	3.646	−0.666	0.581	0.567	0.481
	−0.760	0.132	7.460	8.658	2.254	9.645	1.979	9.527	8.957	0.226	0.427	0.385
	−3.982	6.248	1.166	5.434	8.889	−0.368	1.724	8.892	8.810	0.140	0.261	1.065

Table 3 Computation times in seconds for Example 1

Routine	R1	R2	R3	R1*	R2*	R3*
Example 1	202	97	101	136	66	72

Table 4 Computation times in seconds for Example 2

Routine	R1	R2	R3	R1*	R2*	R3*
Example 2	21	-	63	15	-	47

6 Conclusions

This paper applied interval-analysis methods to solve the direct geometrico-static problem of cable-driven parallel robots with 3 cables. The task consists in finding all equilibrium configurations of the end-effector when the cable lengths are assigned. The problem is challenging, since loop-closure and equilibrium equations must be solved simultaneously.

The algorithm searches for all real solutions within a predetermined domain, whose frontier is computed so as to ensure that all possible solutions are enclosed within. The domain is subdivided into regions. An evaluation operator verifies if a region contains a solution, whereas some filter operators exclude portions which cannot contain roots. Regions whose assessment is uncertain are bisected and further assessed. The code is able to discard solutions in which one or more cables are subject to negative tensile forces.

Interval analysis requires great experience to be implemented with success and choosing the right heuristics has a dramatic impact on the effectiveness of this tool. The evaluation operator adopted in the current version performs a sharper interval evaluation of functions by using gradients and by taking into account possible monotonicities. The filter operators are based on the 2B and 3B consistency methods, which significantly reduce the number of boxes processed and thus the computation time. Interval analysis has a structure that is appropriate for parallel implementation. Accordingly, a distributed version of the algorithm was presented that takes advantage of modern multi-core CPUs. Finally, three equation sets emerging from different formalizations of the static constraints were implemented and a comparison between their main advantages and disadvantages was reported.

The results obtained by the numerical experimentation conducted so far are promising. The code is robust and reliable, and it is able to find all solutions of the problem for a generic geometry within a few minutes. The code has wide margins of improvement, by enhancing the implemented heuristics. In this perspective, a number of refinements were identified that should significantly enhance the computation efficiency and that will be implemented in an upgraded version of the code.

References

[1] Abbasnejad, G., Carricato, M.: Real solutions of the direct geometrico-static problem of under-constrained cable-driven parallel robots with 3 cables: a numerical investigation. Meccanica (2012). DOI 10.1007/s11012-012-9552-3

[2] Carricato, M., Merlet, J.P.: Geometrico-static analysis of under-constrained cable-driven parallel robots. In: J. Lenarčič, M.M. Stanišič (eds.) Advances in Robot Kinematics: Motion in Man and Machine, pp. 309–319. Springer, Dordrecht (2010)

[3] Carricato, M., Merlet, J.P.: Direct geometrico-static problem of under-constrained cable-driven parallel robots with three cables. In: Proc. of the 2011 IEEE Int. Conf. on Robotics and Automation, pp. 3011–3017. Shanghai, China (2011)

[4] Carricato, M., Merlet, J.P.: Inverse geometrico-static problem of under-constrained cable-driven parallel robots with three cables. In: Proc. of the 13th World Congress in Mechanism and Machine Science, pp. 1–10, Paper No. A7_283. Guanajuato, Mexico (2011)

[5] Collard, J.F., Cardou, P.: Computing the lowest equilibrium pose of a cable-suspended rigid body. Optimization and Engineering (2012)

[6] Fattah, A., Agrawal, S.K.: On the design of cable-suspended planar parallel robots. ASME Journal of Mechanical Design **127**(5), 1021–1028 (2006)

[7] Heyden, T., Woernle, C.: Dynamics and flatness-based control of a kinematically undetermined cable suspension manipulator. Multibody System Dynamics **16**(2), 155–177 (2006)

[8] Jansson, C., Rohn, J.: An algorithm for checking regularity of interval matrices. SIAM Journal on Matrix Analysis and Applications **20**(3), 756–776 (1999)

[9] Jaulin, L.: Applied interval analysis: with examples in parameter and state estimation, robust control and robotics, vol. 1. Springer Verlag (2001)

[10] Jiang, Q., Kumar, V.: The direct kinematics of objects suspended from cables. In: Proc. of the ASME 2010 Int. Design Engineering Technical Conferences, pp. 1–10, Paper no. DETC2010–28,036. Montreal, Canada (2010)

[11] McCarthy, J.M.: 21st century kinematics: synthesis, compliance, and tensegrity. ASME Journal of Mechanisms and Robotics **3**(2), 020201 (2011)

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

[13] Merlet, J.P.: ALIAS-C++. `http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS-C++/ALIAS-C++.html` (2007)

[14] Merlet, J.P., Daney, D.: A portable, modular parallel wire crane for rescue operations. In: Proc. of the 2010 IEEE Int. Conf. on Robotics and Automation, pp. 2834–2839. Anchorage, USA (2010)

[15] Michael, N., Kim, S., Fink, J., Kumar, V.: Kinematics and statics of cooperative multi-robot aerial manipulation with cables. In: Proc. of the ASME 2009

Int. Design Engineering Technical Conferences, vol. 7A, pp. 83–91, paper no. DETC2009–87,677. San Diego, USA (2009)

[16] Ming, A., Higuchi, T.: Study on multiple degree-of-freedom positioning mechanism using wires—Part 1: Concept, design and control. Int. Journal of the Japan Society for Precision Engineering **28**(2), 131–138 (1994)

[17] Moore, R., Bierbaum, F.: Methods and applications of interval analysis, vol. 2. Society for Industrial Mathematics (1979)

[18] Rosati, G., Gallina, P., Masiero, S.: Design, implementation and clinical tests of a Wire-Based robot for neurorehabilitation. IEEE Transactions on Neural Systems and Rehabilitation Engineering **15**(4), 560–569 (2007)

[19] Surdilovic, D., Zhang, J., Bernhardt, R.: STRING-MAN: wire-robot technology for safe, flexible and human-friendly gait rehabilitation. In: Proc. of the 2007 IEEE Int. Conference on Rehabilitation Robotics, pp. 446–453. Noordwijk, The Netherlands (2007)

[20] Tapia, R.: The Kantorovitch theorem for Newton's method. American Mathematic Monthly **78**(1.ea), 389–392 (1971)

[21] Yamamoto, M., Yanai, N., Mohri, A.: Trajectory control of incompletely restrained Parallel-Wire-Suspended mechanism based on inverse dynamics. IEEE Transactions on Robotics **20**(5), 840–850 (2004)