# A Tension Distribution Method with Improved Computational Efficiency

Johann Lamaury, Marc Gouttefarde

# A Tension Distribution Method with Improved Computational Efficiency

**Johann Lamaury and Marc Gouttefarde**

**Abstract** This paper introduces a real-time capable tension distribution algorithm for $n$ degree-of-freedom cable-driven parallel robots (CDPR) actuated by $n+2$ cables. It is based on geometric considerations applied to the two-dimensional convex polytope of feasible cable tension distribution. This polytope is defined as the intersection between the set of inequality constraints on the cable tension values and the affine space of tension solutions to the mobile platform static or dynamic equilibrium. The algorithm proposed in this paper is dedicated to $n$ degree-of-freedom CDPR actuated by $n + 2$ cables. Indeed, it takes advantage of the two-dimensional nature of the corresponding feasible tension distribution convex polytope to improve the computational efficiency of a tension distribution strategy proposed elsewhere. The fast computation of the polytope vertices and of its barycenter made us successfully validate the real-time compatibility of the presented algorithm.

## 1 Introduction

A cable-driven parallel robot (CDPR) mainly consists of a base, a mobile platform connected in parallel to the base through flexible cables and motorized winches. The cable lengths can be modified by means of the winches thereby allowing the motion control of the platform. Contrary to common parallel robot architectures, CDPR with large to very large workspaces can be designed as cables can be unwound over great lengths. Moreover, their light weight, fast motion, heavy payload capabilities and high reconfigurability potential make these robots good candidates for large-

J. Lamaury (✉) · M. Gouttefarde
Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier,
161 rue Ada, 34392 Montpellier Cedex 5, France
e-mail: johann.lamaury@lirmm.fr

M. Gouttefarde
e-mail: marc.gouttefarde@lirmm.fr

dimension applications, for example pick-and-place tasks across a manufacturing plant.

CDPR are generally classified as fully constrained or else under-constrained. Examples of fully constrained CDPR are the FALCON robot [6], the SEGESTA [4] and the KNTU [10]. A well-known under-constrained CDPR is the NIST ROBOCRANE [1]. In both cases, the number of cables driving the mobile platform can be greater than its number of degrees of freedom (DOF). When controlling such redundantly actuated CDPR, the issue of cable tension distribution is to be dealt with. Indeed, at any point along a trajectory, there exists an infinity of possible sets of cable tensions and one generally wants to find a feasible one, possibly satisfying some optimality criterion. A cable tension vector is feasible when all its components are contained between minimal and maximal tension values $t_{\min}$ and $t_{\max}$. The maximum $t_{\max}$ is notably given by the maximal admissible cable strain whereas $t_{\min}$ is usually set as the lowest acceptable tension with the goal of avoiding slack cables ($t_{\min} \geq 0$).

Several methods have been proposed, mainly for fully constrained CDPR, in order to find (optimal) feasible tension distributions among the infinite number of possible ones. Fang et al. [4] put forward an optimal tension distribution algorithm that uses a 1-norm linear programming method (LPM) for configuration with one degree of redundancy (DOR), i.e. in the case $m = n + 1$ where $m$ and $n$ are the number of cables and of DOF, respectively. LPM was also used to solve higher DOR in [2, 10]. However, LPM does not guarantee the continuity along a given trajectory which may result in high mechanical loads and vibrations. To avoid them, quadratic programming methods (QPM) may be used [3, 5] but they are suffering from non-predictable worst-case runtime as specified in [3]. For suspended CDPR, Oh and Agrawal [8] proposed to plan the robot trajectory to stay into the feasible tension space by describing this latter as a set of linear inequalities. Yu et al. [11] applied QPM for the control of suspended CDPR with redundant cables, coupling basic tension optimization problem to an active stiffness control scheme.

Nevertheless, all these methods are using optimization procedures, which are most of the time expensive in terms of computation time and against real-time control constraints because of their iterative nature. Consequently, for real-time control needs, a deterministic non-iterative method is highly preferable. Mikelsons et al. [7] proposed such a method in which the barycenter of the *polytope of feasible tension distributions* is determined. To deal with the case of redundant under-constrained CDPR (crane-like configuration), the method proposed in [7] has the additional interest of providing a tension vector contained within the polytope of feasible tension distributions "far" from the polytope boundaries. However, this method requires the computation of all the vertices of this polytope which takes a lot of time if a brute force method, such as the one proposed in [7], is used.

The contribution of this paper is a fast algorithm aiming at a real-time implementation of the barycentric approach proposed in [7]. The proposed algorithm is dedicated to CDPR with two DOR, i.e., actuated by $m = n+2$ cables. Indeed, it takes advantage of the two-dimensional nature of the corresponding polytope of feasible tension distributions which is in fact a convex polygon. The idea consists essentially

in computing a first vertex of this polygon and, then, in finding the others by "moving" from one vertex to the next one while following the polygon one-dimensional boundary which is made of straight line segments. Once all the polygon vertices are determined, the barycenter (centroid) is simply obtained by well-known closed-form formulas. Therefore, in the case of CDPR with two DOR, this paper complements [7] by providing an efficient means of implementing the tension distribution strategy proposed therein.

This paper is organized as follows. Section 2 details the proposed real-time capable algorithm. A brief description of our prototype is given in Sect. 3. Some preliminary simulation results are reported in Sect. 4. Finally, conclusions and future works are addressed in the last section.

## 2 A Fast Tension Distribution Algorithm

### *2.1 Mikelsons' Barycenter Approach*

The wrench $\mathbf{f}$ applied by the cables on the mobile platform is given by [9]

$$\mathbf{W}\mathbf{t} = \mathbf{f} \tag{1}$$

where $\mathbf{W}$ is the wrench matrix and $\mathbf{t} = [t_1, ..., t_m]^T \in \mathbb{R}^m$ is the cable tension vector. The challenge lies in the cable inability to transmit compressive forces, which means that $\mathbf{t}$ has to remain non-negative. This paper deals with $n$-DOF CDPR driven by $m = n + 2$ cables, i.e., with $r = 2$ DOR. Consequently, the $n \times m$ wrench matrix $\mathbf{W}$ is non-square and, assuming that $\mathbf{W}$ has full rank, (1) is equivalent to

$$\mathbf{t} = \mathbf{W}^{+}\mathbf{f} + \mathbf{N}\boldsymbol{\lambda} = \mathbf{t}_p + \mathbf{t}_n \tag{2}$$

where $\mathbf{W}^{+}$ is the Moore-Penrose pseudo-inverse of the wrench matrix, $\mathbf{N} = null(\mathbf{W}) = [\mathbf{k}_1 \ \ \mathbf{k}_2]$ is a full rank $m \times 2$ matrix and $\boldsymbol{\lambda} = [\lambda_1 \ \ \lambda_2]^T$ is an arbitrary 2-dimensional vector. The two columns of $\mathbf{N}$ form a basis of the nullspace of $\mathbf{W}$. $\mathbf{t}_p$ is the particular minimum-norm solution of (1) and $\mathbf{t}_n$ is the homogeneous solution that maps $\lambda$ to the nullspace of $\mathbf{W}$.

Let us define $\Sigma \subset \mathbb{R}^m$ the $r$-dimensional affine space ($r = 2$ in this paper) of the solutions to (1)

$$\Sigma = \{\mathbf{t} \mid \mathbf{W}\mathbf{t} = \mathbf{f}\} \tag{3}$$

whose points are given by (2). Let us also define $\Omega \in \mathbb{R}^m$ as the $m$-dimensional hypercube of feasible cable tensions

$$\Omega = \{\mathbf{t} \mid t_i \epsilon \, [t_{\min}, t_{\max}], \ 1 \leq i \leq m\} \tag{4}$$

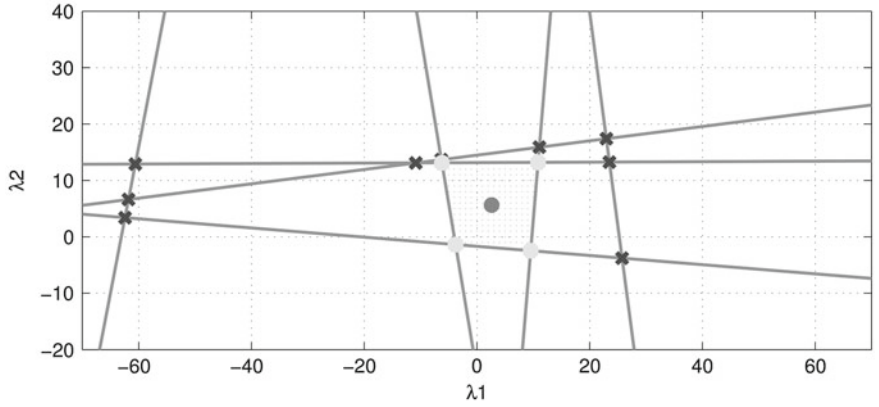


**Fig. 1** Preimage of $\Lambda$ in the plane $(\lambda_1, \lambda_2)$ for the pose $[0.9 \;\; 1.2 \;\; 1 \;\; 10 \;\; 10 \;\; 20]^T$ (units: meters and degrees) with a 6.34 kg total mass (XYZ Euler angle convention)

where we assume that the $t_{\min}$ and $t_{\max}$ limiting tension values are the same for the $m$ cables. The intersection $\Lambda = \Omega \cap \Sigma$ of the hypercube $\Omega$ and the affine space $\Sigma$ is a convex polytope [12]. This *feasible tension distribution polytope* $\Lambda$ represents the set of tension solutions $\mathbf{t}$ to (1) satisfying the inequalities $t_{\min} \leq t_i \leq t_{\max}$. The preimage of $\Lambda$ under the affine mapping $(\mathbf{N}, \mathbf{t}_p)$ is also a convex polytope which, according to (2), is defined by the following set of $2m$ linear inequalities

$$\mathbf{t}_{\min} - \mathbf{t}_p \leq \mathbf{N}\boldsymbol{\lambda} \leq \mathbf{t}_{\max} - \mathbf{t}_p \tag{5}$$

In this paper, since $r = 2$, the feasible tension distribution convex polytope $\Lambda$ is two-dimensional and its preimage under the affine mapping $(\mathbf{N}, \mathbf{t}_p)$ is thus a convex polygon.

For example, Fig. 1 shows the preimage of $\Lambda$ obtained for a static equilibrium pose of the mobile platform of ReelAx8, a CDPR prototype briefly described in Sect. 3. In (5), each of the $2m$ inequalities defines an halfplane. Each of the $2m$ lines bounding these halfplanes corresponds to values of $\boldsymbol{\lambda}$ for which one of the cable tension is equal to $t_{\min}$ or to $t_{\max}$. Figure 1 shows some of these lines together with the preimage of $\Lambda$.

In order to select a "safe" tension distribution, i.e. one which is far from the boundaries of the polytope $\Lambda$, Mikelsons et al. [7] proposed to find $\Lambda$ barycenter. Their method consists essentially in computing all the vertices of the preimage of $\Lambda$. To this end, all the $2 \times 2$ subsystems of linear equations obtained by selecting two of the $2m$ inequalities of (5) are solved. A solution $\boldsymbol{\lambda}$ of one of these systems is a vertex if it verifies (5). Once all the vertices of the preimage of $\Lambda$ are known, in [7], its barycenter $\boldsymbol{\lambda}_c$ is determined by means of a triangulation. Finally, the barycenter of $\Lambda$ is calculated as the image of $\boldsymbol{\lambda}_c$ under the affine mapping $(\mathbf{N}, \mathbf{t}_p)$, i.e., as $\mathbf{t}_p + \mathbf{N}\boldsymbol{\lambda}_c$.

The computational burden of the method presented in [7] is thus mainly determined by the number of non-singular $2 \times 2$ subsystems of linear equations drawn from (5). In the case $n = 6$ and $m = 8$, there are $C_{16}^2 = 120$ such subsystems. But, as noted in [2], since each line of (5) defines two halfplanes bounded by two parallel lines, the number of non-singular ones is equal to $C_{16}^2 - 8 = 112$.

The method of Mikelsons et al. provides a continuous tension distribution with a predictable worst-case runtime. However, because of the high number of linear systems that must be solved, the computation time is too high to fit our real-time controller loop time so that we were not able to implement it on our prototype ReelAx8. Besides, to the best of our knowledge, no real-time implementation of this method has been reported.

The present paper proposes a strong decrease of the computational time of this method by finding a first vertex of the convex polytope $\Lambda$ and then moving along the polytope hull in order to determine the other vertices. An efficient implementation of this idea is discussed in Sect. 2.2. It is dedicated to CDPR with 2 DOR as it requires the feasible tension distribution convex polytope to be two dimensional. Simulation results show that the proposed improvement provides a real-time compatibility.

## *2.2 Detailed Description of the Proposed Algorithm*

### 2.2.1 Initialization

Let us consider the intersection point, represented by the two-dimensional column vector $\boldsymbol{\lambda}_{ij}$, between two lines $L_{i_b}$ and $L_{j_b}$, $\{i, j\} \in \{1, \ldots, m\}, i \neq j$. $L_{i_b}$ and $L_{j_b}$ are obtained by taking two inequalities among the $2m$ of (5) and replacing the inequality signs by equalities. These two lines are thus defined by the following equations

$$\begin{cases} b_i - t_{p_i} = \mathbf{n}_i \boldsymbol{\lambda}_{ij} \\ b_j - t_{p_j} = \mathbf{n}_j \boldsymbol{\lambda}_{ij} \end{cases} \tag{6}$$

where each one of $b_i$ and $b_j$ is equal either to $t_{\min}$ or to $t_{\max}$ depending on which inequalities are being considered. The two-dimensional line vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ denote the lines $i$ and $j$ of $\mathbf{N}$, respectively. Examples of such $L_{i_b}$ and $L_{j_b}$ lines and intersection points $\boldsymbol{\lambda}_{ij}$ can be seen in Fig. 1. Furthermore, $\boldsymbol{\lambda}_{ij}$ is a vertex $\mathbf{v}_{ij}$ of the preimage of $\Lambda$ if

$$\mathbf{t}_{\min} - \mathbf{t}_p \leq \mathbf{N}\boldsymbol{\lambda}_{ij} \leq \mathbf{t}_{\max} - \mathbf{t}_p \tag{7}$$

which means that $\boldsymbol{\lambda}_{ij}$ is included into $\Omega$. The algorithm proposed in this paper consists in first finding a vertex $\mathbf{v}_{init}$ of the preimage of $\Lambda$, which is a convex polygon, and, then, in *moving* along one of the two lines $L_{i_b}$ or $L_{j_b}$ intersecting at $\mathbf{v}_{init}$ until a new vertex of the polygon is reached. This process continues until it reaches the polygon

vertex which belongs to the other line intersecting at $\mathbf{v}_{init}$, i.e, when all the vertices of the convex polygon have been found.

The first step is thus the calculation of the first vertex $\mathbf{v}_{init}$. To this end, we are looking for an intersection point $\boldsymbol{\lambda}_{ij}$ which satisfies the following requirements

$$\begin{cases} b_i - t_{p_i} = \mathbf{n}_i \boldsymbol{\lambda}_{ij} \\ b_j - t_{p_j} = \mathbf{n}_j \boldsymbol{\lambda}_{ij} \\ \mathbf{t}_{\min} - \mathbf{t}_p \leq \mathbf{N}\boldsymbol{\lambda}_{ij} \leq \mathbf{t}_{\max} - \mathbf{t}_p \end{cases} \tag{8}$$

By taking a couple $i$ and $j$ of cables among the $C_m^2$ available and taking $t_{\min}$ or $t_{\max}$ as the value of $b_i$ and of $b_j$, the two equalities of (8) are solved (if the corresponding linear system is not singular), and the resulting vector $\boldsymbol{\lambda}_{ij}$ is the searched $\mathbf{v}_{init}$ vertex if the two inequalities of the last line of (8) are verified. From a general point of view, it may be necessary to consider many couples $i$ and $j$ of cables and combinations of $t_{\min}$ and $t_{\max}$ values before such a first vertex of the polygon can be found. However, in practice, let us note that the computation of the first vertex $\mathbf{v}_{init}$ should generally not be an issue since the first point of a trajectory is generally a known (static equilibrium) mobile platform pose at which the preimage of $\Lambda$ is already determined. Indeed, this first pose is typically either the home starting pose of the CDPR for which all the computations can be done offline and once and for all, or else the end point of a previous trajectory for which the preimage of $\Lambda$ has been calculated previously. For any other point of the trajectory at hand, since the preimage of $\Lambda$ evolves continuously in time, the first vertex $\mathbf{v}_{\text{init}}$ is easily obtained from the first vertex or any of the other vertices of preimage of $\Lambda$ associated with the previous point of the trajectory.

### 2.2.2 From One Vertex to the Next One

Once the first vertex $\mathbf{v}_{init}$ is known, the second step consists in moving along one of the two lines $L_{i_b}$ or $L_{j_b}$ intersecting at $\mathbf{v}_{init}$ until a new vertex $\mathbf{v}$ of the convex polygon (preimage of $\Lambda$) is found. Let us arbitrarily choose $L_{i_b}$. The points $\mathbf{p}$ belonging to this line are given by

$$\mathbf{p} = \mathbf{v}_{init} + \alpha \mathbf{n}_{i_\perp}^T \tag{9}$$

where $\alpha$ is a scalar and $\mathbf{n}_i \cdot \mathbf{n}_{i_\perp}^T = 0$, i.e., the line vector $\mathbf{n}_{i_\perp}$ is orthogonal to $\mathbf{n}_i$ and thus defines the direction of the line $L_{i_b}$. With $\mathbf{n}_i = [a \;\; b]$, there exists two possible vectors $\mathbf{n}_{i_\perp}$ which are $\mathbf{n}_{i_{\perp 1}} = [b \;\; -a]$ and $\mathbf{n}_{i_{\perp 2}} = [-b \;\; a]$. Care must be taken in the choice between these two possible vectors. Indeed, the goal is to move along the boundary of the convex polygon and not to follow $L_{i_b}$ while moving away from the polygon. Let us decide that $\alpha \geq 0$ in (9) so that $\mathbf{n}_{i_\perp}$ must be directed toward the interior of the polygon in order to move along the polygon boundary. Two cases have to be distinguished.

- Case 1: $b_j = t_{\min}$. In this case, moving along line $L_{i_b}$ from its intersection point with line $L_{j_b}$ while staying on the polygon boundary requires

$$\begin{array}{c} t_{\min} - t_{p_j} \leq \mathbf{n}_j \mathbf{p} \Leftrightarrow t_{\min} - t_{p_j} \leq \mathbf{n}_j \mathbf{v}_{init} + \alpha \mathbf{n}_j \mathbf{n}_{i_\perp}^T \\ \Leftrightarrow \qquad\qquad \mathbf{n}_j \mathbf{n}_{i_\perp}^T \geq 0 \end{array} \tag{10}$$

  The last equivalence is true because $\alpha \geq 0$ and also because, $\mathbf{v}_{init}$ lying on line $L_{j_b}$, we have $t_{\min} - t_{p_j} = \mathbf{n}_j \mathbf{v}_{init}$. Therefore, among the two possible vectors $\mathbf{n}_{i_\perp}$, the good choice in order to stay on the polygon boundary is the one such that $\mathbf{n}_j \mathbf{n}_{i_\perp}^T \geq 0$.

- Case 2: $b_j = t_{\max}$. In this second case, moving along line $L_{i_b}$ from its intersection point with line $L_{j_b}$ while staying on the polygon boundary requires

$$\begin{array}{c} \mathbf{n}_j \mathbf{p} \leq t_{\max} - t_{p_j} \Leftrightarrow \mathbf{n}_j \mathbf{v}_{init} + \alpha \mathbf{n}_j \mathbf{n}_{i_\perp}^T \leq t_{\max} - t_{p_j} \\ \Leftrightarrow \qquad\qquad \mathbf{n}_j \mathbf{n}_{i_\perp}^T \leq 0 \end{array} \tag{11}$$

  since $\alpha \geq 0$ and $\mathbf{n}_j \mathbf{v}_{init} = t_{\max} - t_{p_j}$. This time, among the two possible vectors $\mathbf{n}_{i_\perp}$, the good choice is the one such that $\mathbf{n}_j \mathbf{n}_{i_\perp}^T \leq 0$.

Now that we know along which direction to move along line $L_{i_b}$ in order to follow the polygon boundary from vertex $\mathbf{v}_{init}$, we aim at finding the other polygon vertex $\mathbf{v}$ belonging to $L_{i_b}$. In fact, this other vertex is the point $\mathbf{p}$ in (9) corresponding to the maximal value of $\alpha \geq 0$ such that all the inequalities of (5) are verified. Equivalently, this maximal value is equal to the smallest $\alpha \geq 0$ such that one of the inequalities of (5) apart from inequalities $i$ and $j$ becomes an equality. Therefore, let us consider line $k$ of (5), $k \in \{1, \ldots, m\} \backslash \{i, j\}$, and let us substitute $\boldsymbol{\lambda}$ by the point $\mathbf{p}$ of $L_{i_b}$ given by (9), i.e.

$$\begin{cases} t_{\min} - t_{p_k} \leq \mathbf{n}_k \mathbf{p} \leq t_{\max} - t_{p_k} \\ \mathbf{p} = \mathbf{v}_{init} + \alpha \mathbf{n}_{i_\perp}^T \end{cases} \tag{12}$$

which is equivalent to

$$t_{\min} - t_{p_k} \leq \mathbf{n}_k \mathbf{v}_{init} + \alpha \mathbf{n}_k \mathbf{n}_{i_\perp}^T \leq t_{\max} - t_{p_k} \tag{13}$$

Let us assume that none of the two $L_{k_b}$ lines, the two lines bounding the halfplanes defined by the two inequalities of line $k$ of (5), crosses line $L_{i_b}$ at $\mathbf{v}_{init}$. This amounts to assuming that

$$t_{\min} - t_{p_k} < \mathbf{n}_k \mathbf{v}_{init} < t_{\max} - t_{p_k} \tag{14}$$

The particular case in which three lines are crossing at the same point (here lines $L_{i_b}$, $L_{j_b}$ and $L_{k_b}$ crossing at the current vertex) is addressed in Sect. 2.3.

Now, let us consider (13) and (14).

- If $\mathbf{n}_k \mathbf{n}_{i_\perp}^T = 0$ then $\forall \alpha \geq 0$ the inequalities of the system (12) cannot become equalities (which means that $L_{i_b}$ is parallel to the two $L_{k_b}$ lines).
- If $\mathbf{n}_k \mathbf{n}_{i_\perp}^T > 0$ then $\forall \alpha \geq 0$,

$$\mathbf{n}_k \mathbf{v}_{init} + \alpha \mathbf{n}_k \mathbf{n}_{i_\perp}^T \geq \mathbf{n}_k \mathbf{v}_{init} > t_{\min} - t_{p_k} \tag{15}$$

Consequently, (13) could only become an equality by its right side when $\alpha \geq 0$, i.e., $\mathbf{n}_k \mathbf{v}_{init} + \alpha \mathbf{n}_k \mathbf{n}_{i_\perp}^T = t_{\max} - t_{p_k}$, which is satisfied by the following value of $\alpha \geq 0$

$$\alpha_k = \frac{t_{\max} - t_{p_k} - \mathbf{n}_k \mathbf{v}_{init}}{\mathbf{n}_k \mathbf{n}_{i_\perp}^T} \tag{16}$$

- If $\mathbf{n}_k \mathbf{n}_{i_\perp}^T < 0$ then $\forall \alpha \geq 0$,

$$\mathbf{n}_k \mathbf{v}_{init} + \alpha \mathbf{n}_k \mathbf{n}_{i_\perp}^T \leq \mathbf{n}_k \mathbf{v}_{init} < t_{\max} - t_{p_k} \tag{17}$$

Consequently, (13) could only become an equality by its left side when $\alpha \geq 0$, i.e., $\mathbf{n}_k \mathbf{v}_{init} + \alpha \mathbf{n}_k \mathbf{n}_{i_\perp}^T = t_{\min} - t_{p_k}$, which is satisfied by

$$\alpha_k = \frac{t_{\min} - t_{p_k} - \mathbf{n}_k \mathbf{v}_{init}}{\mathbf{n}_k \mathbf{n}_{i_\perp}^T} \tag{18}$$

As a consequence, the sign $s_k = Sign(\mathbf{n}_k \mathbf{n}_{i_\perp}^T)$ indicates which side of inequality (13) can become an equality for $\alpha \geq 0$. Moreover, since $\mathbf{v}_{init}$ is the intersection of lines $L_{i_b}$ and $L_{j_b}$, the line $L_{k_b}$ must be found for $k$ belonging to the index set $\{1, \ldots, m\} \backslash \{i, j\}$ that leaves $2(m-2)$ possibilities (two possible line equations per line of (5)). By computing $s_k$, the previous analysis allows us to reduce this number to $m-2$ possibilities.

In order to find the next vertex of the polygon, i.e., the second polygon vertex belonging to $L_{i_b}$, the $m$ $\alpha_k$ are thus computed by means of (16) or (18) and the one, denoted $\alpha_v$, which will determine the next vertex is the smallest, that is

$$\alpha_v = \min_{k, \mathbf{n}_k \mathbf{n}_{i_\perp}^T \neq 0} \left( \frac{b_k - t_{p_k} - \mathbf{n}_k \mathbf{v}_{init}}{\mathbf{n}_k \mathbf{n}_{i_\perp}^T} \right) \tag{19}$$

where $b_k = t_{\min}$ or $t_{\max}$ despite of the value of $s_k$. The next vertex $\mathbf{v}$ is thus given by

$$\mathbf{v} = \mathbf{v}_{init} + \alpha_v \mathbf{n}_{i_\perp}^T \tag{20}$$

and the line $L_{l_b}$ which crosses $L_{i_b}$ at $\mathbf{v}$ while supporting the polygon along one of its edge is the set of point $\mathbf{p}$ verifying $b_l - t_{p_l} = \mathbf{n}_l \mathbf{p}$ where $b_l = t_{\max}$ if $\mathbf{n}_l \mathbf{n}_{i_\perp}^T > 0$ and $b_l = t_{\min}$ if $\mathbf{n}_l \mathbf{n}_{i_\perp}^T < 0$ and

$$l = \underset{k, \mathbf{n}_k \mathbf{n}_{i_\perp}^T \neq 0}{\operatorname{argmin}} \left( \frac{b_k - t_{p_k} - \mathbf{n}_k \mathbf{v}_{init}}{\mathbf{n}_k \mathbf{n}_{i_\perp}^T} \right) \tag{21}$$

Starting from this newly found polygon vertex $\mathbf{v}$, which is the intersection point between $L_{i_b}$ and $L_{l_b}$, and moving along line $L_{l_b}$ (in the appropriate direction), the next polygon vertex is found in exactly the same way as vertex $\mathbf{v}$ has been found. This process continues until the newly found vertex lies on the same line as $\mathbf{v}_{init}$ ($L_{j_b}$ in our example), at which point the research stops since all the vertices of the convex polygon, preimage of $\Lambda$, have been determined.

### 2.2.3 The Barycenter Calculation

The final step is the calculation of the barycenter $\mathbf{v}_c$ of the preimage of $\Lambda$ whose $q$ vertices $\mathbf{v}_p = [v_{p_1} \; v_{p_2}]^T$, $p \in \{1, \ldots, q\}$ have just all been determined. The preimage of $\Lambda$ is a convex polygon which is not self-intersecting. Therefore, its centroid $\mathbf{v}_c = [v_{c_1} \; v_{c_2}]^T$ is given by the following well-known formulas

$$\begin{cases} v_{c_1} = \frac{1}{6A} \sum_{p=0}^{q-1} (v_{p_1} + v_{(p+1)_1})(v_{p_1} v_{(p+1)_2} - v_{(p+1)_1} v_{p_2}) \\ v_{c_2} = \frac{1}{6A} \sum_{p=0}^{q-1} (v_{p_2} + v_{(p+1)_2})(v_{p_1} v_{(p+1)_2} - v_{(p+1)_1} v_{p_2}) \end{cases} \tag{22}$$

where $A$ is the area of the polygon given by

$$A = \frac{1}{2} \sum_{p=0}^{q-1} (v_{p_1} v_{(p+1)_2} - v_{(p+1)_1} v_{p_2}) \tag{23}$$

Finally, the polygon barycenter $\mathbf{v}_c$ is mapped into $\mathbb{R}^m$ in order to find $\mathbf{t}_c = \mathbf{t}_p + \mathbf{N}\mathbf{v}_c$, the barycenter of $\Lambda$ which is the desired feasible cable tension distribution.

As an example, Fig. 1 shows $\mathbf{v}_c$ as the dark dot into the polygon. The points indicated with crosses correspond to some intersection points between lines bounding the halfplanes defined in (5). The four clear dots are the vertices of the preimage of $\Lambda$.

## *2.3 Case of Three (Ore More) Concurrent Lines*

In the algorithm proposed in Sect. 2.2, let us assume that we have to move from a vertex $\mathbf{v}_{ij}$ to the next one $\mathbf{v}_{jk}$ along line $j$. In order to compute $\mathbf{v}_{jk}$, the algorithm calculates $m - 2$ values of $\alpha$ as given by (16) or (18) and retains the minimal one. If this minimal value of $\alpha$ is obtained for two indices $k$ and $l$ [i.e., both $k$ and $l$

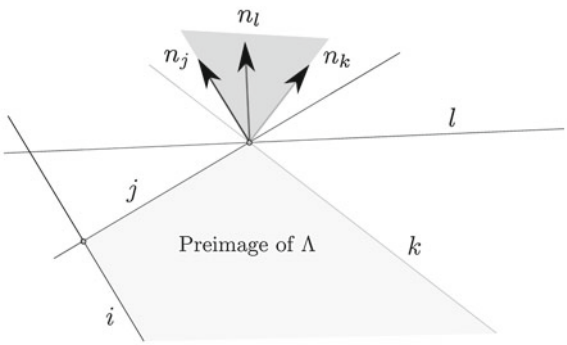


**Fig. 2** Illustration of the case of *three concurrent lines*

verify (21)], it means that the vertex $\mathbf{v}_{jk}$ is the intersection of three concurrent lines (here, lines $j$, $k$ and $l$). This particular case is illustrated in Fig. 2. In such a case, the determination of the vertex $\mathbf{v}_{jk}$ is not a problem but care must be taken in the determination of the next line that will be followed in order to find a new polygon vertex. Indeed, this next line has to support the polygon along one of its edges and not only at $\mathbf{v}_{jk}$. As illustrated in Fig. 2, moving from $\mathbf{v}_{jk}$ along line $l$ instead of line $k$ is a bad choice as we then leave the polygon boundary. Consequently, the algorithm must be able to select the edge supporting line which is either line $k$ or else line $l$. As along a trajectory new polygon vertices are generally coming with this particular case, it should be addressed in order to ensure robustness of the algorithm.

In order to select the appropriate edge supporting line, a simple geometric analysis can be used. Vector $\mathbf{n}_j$ is normal to line $j$ and chosen to be directed outward of the polygon. In our current implementation of the algorithm detailed in Sect. 2.2, as soon as two equal $\alpha$ are calculated, say $\alpha_k$ and $\alpha_l$, the vectors $\mathbf{n}_k$ and $\mathbf{n}_l$ are drawn from lines $k$ and $l$ of $\mathbf{N}$ and chosen to be directed outward of the polygon. Then, we check if $\mathbf{n}_l$ is included in the cone spanned by $\mathbf{n}_j$ and $\mathbf{n}_k$ as shown in Fig. 2. If $\mathbf{n}_l$ is strictly included inside this cone, line $l$ supports the polygon at a vertex only. It is thus discarded and line $k$ is selected as the next line to follow since it is an edge supporting line of the polygon. This is the case illustrated in Fig. 2. Otherwise, when $\mathbf{n}_l$ is not included inside the cone spanned by $\mathbf{n}_j$ and $\mathbf{n}_k$, line $l$ supports the polygon along one of its edge and it will be the next line followed in the quest of a new vertex. Note that in the very particular case in which $\mathbf{n}_l$ and $\mathbf{n}_k$ are collinear, any of the two lines $k$ or $l$ can be followed.

Finally, in the cases in which more than three lines, say $h > 3$ lines, are all concurrent at the same vertex of the polygon, the determination of the edge supporting line to be followed during the next algorithm step can be done by sequentially considering sets of three lines among the $h$ concurrent ones and, for each such set, discarding one of the three lines as explained in the previous paragraph.

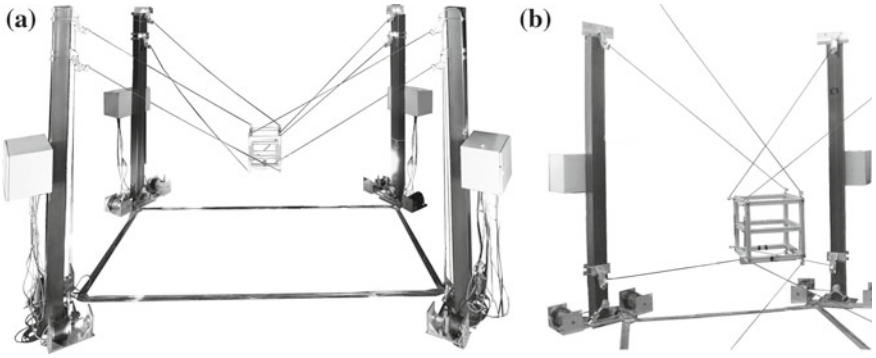


**Fig. 3** The prototype ReelAx8 in suspended under constrained (**a**) and fully constrained (**b**) configurations

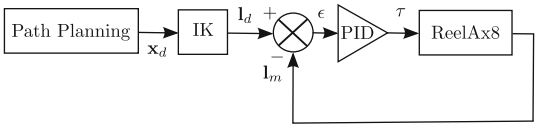


**Fig. 4** Suspended ReelAx8 basic PID-control scheme

## 3 ReelAx8 Prototype

The prototype ReelAx8 has 6-DOF and is driven by 8 cables so that it is *redundantly actuated* with two 2 extra cables (2 DOR). These two extra cables significantly improve the ratio between the workspace and the whole robot footprint. The workspace is defined as the set of feasible static equilibrium mobile platform poses. A pose is said to be feasible when there exists a set of non-negative cable tensions satisfying the platform equilibrium and when there are no cable interferences. Two different configurations have been experimented: suspended and fully constrained, shown in Fig. 3a and b, respectively.

Let us note that redundantly actuated suspended (under constrained) CDPR have rarely been studied [8, 11]. However, this configuration may be required in some situations as all the cable drawing points are located above the workspace. Consequently, the space located below the mobile platform is free of cables. Suspended CDPR have thus the potential of operating in presence of human workers and good transits and are suitable for crane-like applications. ReelAx8 is currently set in its suspended configuration shown in Fig. 3a. As we have not yet implemented an effective control scheme able to use tension distribution strategies for this suspended configuration, the next section gives some preliminary simulation results. Therefore, the real-time compatibility has been established by executing on the real-time controller the algorithm proposed in Sect. 2.2 in parallel of the basic articular PID control scheme, shown in Fig. 4, presently in used on ReelAx8.

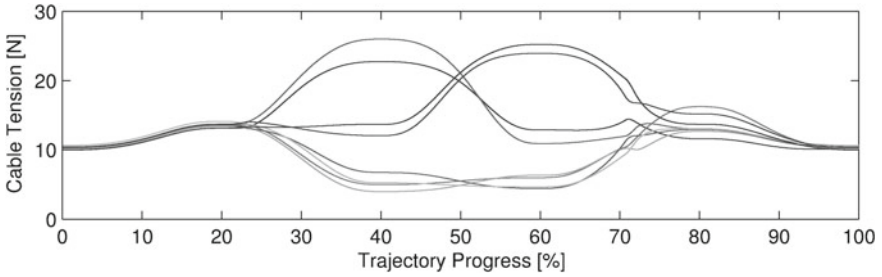


**Fig. 5** Cable tensions obtained with the proposed algorithm for ReelAx8 in suspended configuration

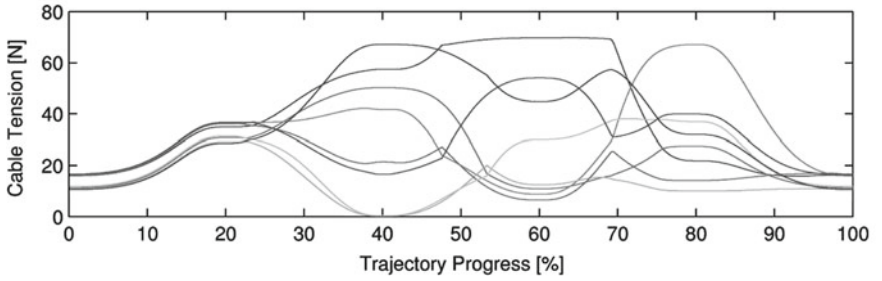


**Fig. 6** Cable tensions obtained with the proposed algorithm for ReelAx8 in fully constrained configuration

## 4 Simulation Results

Figure 5 shows the evolution of the tension into the cables, obtained by means of the algorithm introduced in Sect. 2.2, of the suspended ReelAx8 (Fig. 3a) for a given trajectory of the mobile platform. The trajectory starts at the platform reference pose $[0\ 0\ 0.25\ 0\ 0\ 0]^T$, goes up to $[0.8\ (-0.9)\ 1\ 0\ 0\ 0]^T$, passes through $[0.9\ 0.9\ 1\ 0\ 0\ 0]^T$ and $[0\ 0\ 1\ 0\ 0\ 45]^T$, where a non-zero orientation is accomplished, and finally returns to the reference pose (units are meters and degrees; XYZ Euler angle convention is used to define the platform orientation). Figure 6 shows the evolution of the cable tensions of ReelAx8 in fully constrained configuration (Fig. 3b) along the same trajectory. As observed, all cables are tensed along the whole trajectory and the tension curves are continuous. Let us note that in fully constrained configuration, the choice of the polytope centroid may results in high tension values.

In MATLAB simulations along the same trajectory, the use of the brute force method suggested in [7] which consists in computing all the intersection points between all the lines drawn from (5) results in an average computation runtime of 1.9152 ms against 0.638 ms with our algorithm. This significant improvement should make our algorithm suitable for a future real-time use on ReelAx8.

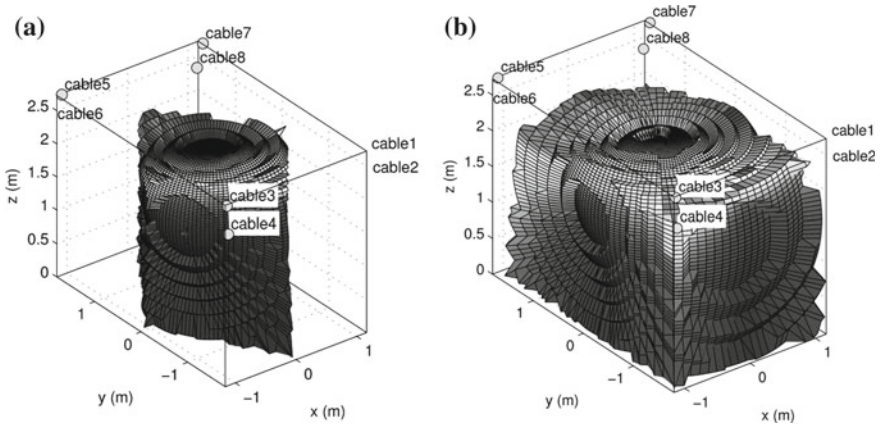


**Fig. 7** Workable part of the constant-orientation static workspace of ReelAx8 without and with a tension distribution algorithm. **a** Without any tension distribution. **b** With the barycenter tension distribution strategy

Furthermore, in order to apply the proposed method experimentally on ReelAx8, the currently used basic control scheme has to be improved. Meanwhile, the algorithm of Sect. 2.2 has been compiled and tested in real-time in parallel of our current control scheme on a target computer using MATLAB/Simulink language. It appeared to be real-time compatible with an average Task Execution Time of 0.156 ms.

Let us note that, compared to the basic PID control of Fig. 4, which outputs negative tension set points in a large part of ReelAx8 workspace, the barycenter tension distribution strategy should increase the practical workspace of ReelAx8 in suspended configuration from 7.58 to 18.9 $m^3$, i.e. 78.18 % of the overall robot occupied volume, as shown in Fig. 7b. These values are obtained for the unloaded platform mass which is of 6.34 kg. The influence of the total mass (loaded platform) on the workspace volume is depicted in Fig. 8. The maximum cable tension considers in this figure is 350 N.

## 5 Conclusions and Future Works

A real-time capable algorithm for tension distribution of cable-driven parallel robots was presented in this paper. This algorithm efficiently implements the barycenter approach of [7] which leads to safe and continuous cable tension distribution. Simulation results showed that it is faster than a brute force implementation of the barycenter approach. The proposed algorithm is dedicated to $n$-DOF parallel robots driven by $n + 2$ cables as it takes advantage of the 2-dimensional nature of the polytope of feasible tension distributions.

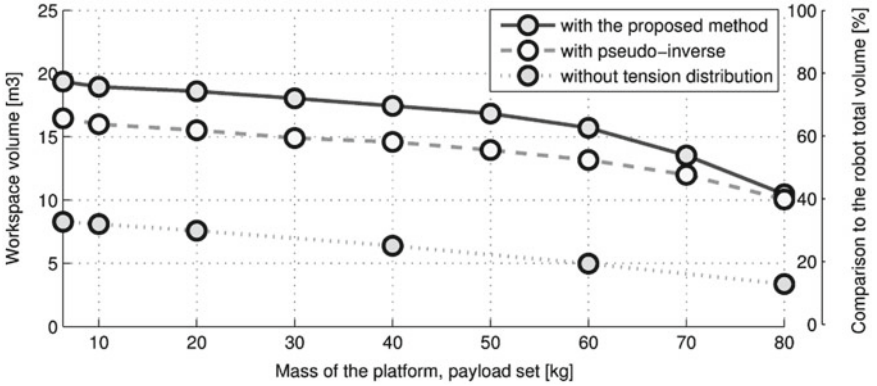


**Fig. 8** Influence of the platform and payload total mass on the suspended ReelAx8 workable workspace volume

We believe that the barycenter cable tension distribution approach is appropriate to deal with the case of redundantly actuated suspended (under constrained) cable-driven parallel robots since it provides a tension distribution set point far from the boundaries of the polytope of feasible tension distributions. However, in the case of fully constrained robots, using the centroid of this polytope as the desired cable tension distribution might not be the better choice as it can lead to large cable tensions and consequently limit the robot workspace and leads to high energy consumption.

The proposed algorithm, tested in real-time on our embedded computer in parallel of our current basic control scheme, must now be implemented within a suitable control scheme. The realization of this latter is part of our future works.

**Acknowledgments** The financial support of the ANR (grant 2009 SEGI 018 01), of the Région Languedoc-Roussillon (grants 115217 and 120218) and the financial contribution of Tecnalia are greatly acknowledged.

## References

1. Albus, J., Bostelman, R., Dagalakis, N.: The NIST robocrane. J. Robot. Syst. **10**(2), 709–724 (1993)
2. Borgstrom, P.H., Jordan, B.L., Sukhatme, G.S., Batalin, M.A., Kaiser, W.J.: Rapid computation of optimally safe tension distributions for parallel cable-driven robots. IEEE Trans. Robot. **25**(6), 1271–1281 (2009)
3. Bruckmann, T., Andreas, P., Hiller, M., Franitza, D.: A modular controller for redundantly actuated tendon-based stewart platforms. EuCoMeS, Obergurgl, Austria, In (2006)
4. Fang, S., Franitza, D., Torlo, M., Bekes, F., Hiller, M.: Motion control of a tendon-based parallel manipulator using optimal tension distribution. IEEE/ASME Trans. Mechatron. **9**, 561–568 (2004)
5. Hassan, M., Khajepour, A.: Analysis of bounded cable tensions in cable-actuated parallel manipulators. IEEE Trans. Robot. **27**(5), 891–900 (2011)

6. Kawamura, S., Kino, H., Won, C.: High-speed manipulation by using parallel wire-driven robots. Robotica **18**, 13–21 (2000)
7. Mikelsons, L., Bruckmann, T., Hiller, M., Schramm, D.: A real-time capable force calculation algorithm for redundant tendon-based parallel manipulators. IEEE International Conference on Robotics and Automation, May, In (2008)
8. Oh, S.-R., Agrawal, S.K.: Controllers with positive cable tensions. IEEE International Conference on Robotics and Automation, Cable-Suspended Planar Parallel Robots with Redundant Cables, In (2003)
9. Roberts, R.G., Graham, T., Lippitt, T.: On the inverse kinematics, statics, and fault tolerance of cable-suspended robots. J. Robot. Syst. **15**(10), 581–597 (1998)
10. Vafaei, A., Aref, M.M., Taghirad, H.D.: Integrated controller for an over constrained cable driven parallel manipulator: KNTU CDRPM. In IEEE 2010 International Conference on Robotics and Automation, pp. 650–655. Anchorage, Alaska, USA (2010).
11. Yu, K., Lee, L., Krovi, V.N.: Simultaneous trajectory tracking and stiffness control of cable actuated parallel manipulator. International Design Engineering Technical Conference and Computers and Information in Engineering Conference, In (2009)
12. Ziegler, G.: Letures on Polytopes. Springer, Heidelberg (1994)