An improved design algorithm based on interval analysis for spatial parallel manipulator with specified workspace

Jean-Pierre MERLET
INRIA Sophia-Antipolis
BP 93 06902 Sophia-Antipolis, France
E-mail: Jean-Pierre.Merlet@sophia.inria.fr

Abstract

We propose an algorithm that enable one to determine almost all the geometries of a simplified Gough platform whose workspace must include an arbitrary set of poses. Five design parameters have been identified and we assume that the stroke of the linear actuator is known. This algorithm is based on an interval analysis approach and its result is a set of ranges for the design parameters such that the workspace of any robot whose geometry is defined by values within the ranges will include the specified poses.

1 Introduction

Parallel robots are becoming more and more attractive for different industrial applications. However one of their drawbacks is that their performances are heavily dependant upon their geometry. Hence optimal design for parallel manipulator is a very important issue. Optimal design has been mainly considered from the viewpoint of isotropy [1, 2, 3]. But one of the main criterion that has to be considered in the design phase is the workspace of the robot as most of the end-user are usually able to specify a desired workspace. Design with respect of a workspace has been considered when the orientation is constant [4] or for planar robot [5] or other specific robots [6],[7],[8]: but in these works only one solution was sought while we will be interested here in finding **almost all** the possible geometries that include a given workspace.

A pose of the platform will be specified by the coordinates (x, y, z) in the reference frame $O, (x, y, z)$ of the origin C of the platform frame and by the pitch,roll, yaw angles ϕ_1, ϕ_2, ϕ_3 (figure 1).

1.1 Design parameters

In this paper we consider a 6 DoF Gough platform having planar base and platform that we will call a *SSM*. The geometry of the base and platform are defined by six variables, namely R_1, r_1, α, β, γ_1, γ_2, where R_1, r_1 are the radii of the circles on which lie the attachment points of the legs A_i, B_i respectively on the base and on the platform, α, β are half the angles between two close attachments points respectively on the base and platform and γ_1, γ_2 are respectively the angles between the middle line between the first and second attachment points and the x, x_r axis (figure 1). The leg geometry is defined by its length

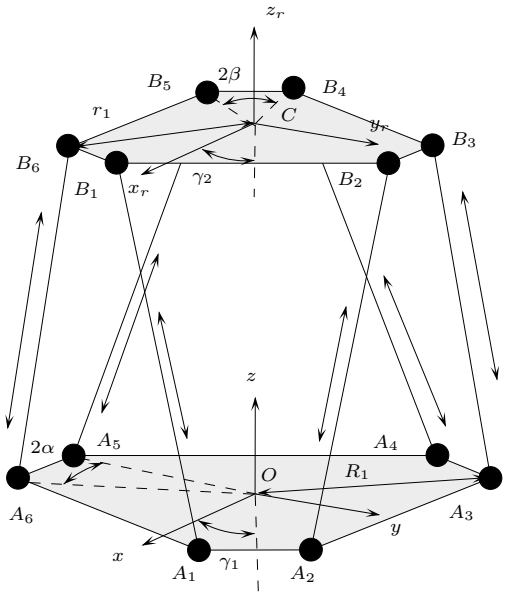

Figure 1: A Gough platform with its design parameters

ρ_d when the actuator is fully retracted, that will be

supposed identical for all legs, and the stroke S of the actuator. Only a few set of possible strokes are available for commercial linear actuators and hence we will assume that the stroke has been chosen beforehand. Thereby the robot geometry is fully defined by 7 parameters. If we furthermore assume that the pose (called the *nominal position*) obtained when the actuators are at their half stroke should be such that $x = 0$, $y = 0$, $\phi_1 = \phi_2 = \phi_3 = 0$, then we must have $\gamma_1 = \pi/6, \gamma_2 = -\pi/6$ and the number of design parameters is reduced to 5: R_1, r_1, α, β and ρ_d. We may notice that this set of parameters is an extension of our previous work [9] in which the value of α, β, ρ_d were supposed to be known.

1.2 Workspace

A set $\mathcal{P}$ of n poses $P_1, \ldots, P_n$ will be specified as the desired poses, the value of n being arbitrary. But a difficulty occurs when we have to define the location of C for each of these poses. The coordinates x, y of C are naturally defined with respect to the value 0 (the coordinates of C in its nominal position) but the z coordinates z_n of the nominal position is dependent on the value of ρ_d. Hence two possibilities will be given for defining the z coordinate of the poses: either its value with respect to the reference frame or a value Δz relative to z_n: in this case the real z of a pose will be $z_n + \Delta z$. Note that the value of z_n may be computed directly from the design parameters as:

$$z_n = \sqrt{-R_1^2 - r_1^2 + 2R_1 r_1 \sin(\frac{\pi}{6} + \alpha + \beta) + (\rho_d + S/2)^2}$$

2 The $\mathcal{A}$ algorithm

To determine if a given pose lie in the workspace of a given robot it is necessary to verify that the lengths ρ_i of the leg corresponding to this pose verify:

$$\rho_{min} = \rho_d \leq \rho_i \leq \rho_d + S = \rho_{max} \qquad (1)$$

for all 6 legs. The lengths ρ_i are clearly a function of the pose parameters, but also of the design parameters. We will see later on that in our algorithm it will be necessary to determine if the previous equation is verified when the design parameters have not a fixed value but may have an arbitrary value within some ranges. We will denote by $\hat{X}$ a range for a design parameter, by $\underline{X}$ the lower bound of the range and by $\overline{X}$ its upper bound: hence $\hat{R_1}$ denote a range for the design parameter R_1 with $\hat{R1} = [\underline{R_1}, \overline{R_1}]$. The width of a range is defined as $\overline{X} - \underline{X}$. The set of 5 ranges for the 5 design parameters will be denoted $\mathcal{D}$ and will be called a *design parameters set*. To verify if equation (1) is satisfied we will compute two quantities $\underline{\rho}, \overline{\rho}$ such that for any value of the design parameters in their ranges we have:

$$\underline{\rho} \leq \rho_i \leq \overline{\rho}$$

In other words we will compute an upper and lower bound for ρ_i. These quantities may be over-estimated (i.e. there is no fixed value of the design parameters such that $\rho_i = \underline{\rho}$ or $\rho_i = \overline{\rho}$). The only constraint is that if the design parameters have a constant value, then $\underline{\rho_i} = \rho_i = \overline{\rho}$ up to round-off errors. The $\mathcal{A}(\mathcal{D}, P_i)$ algorithm applied on a set of design parameters $\mathcal{D}$ and on a pose P_i computes the values of $\underline{\rho}, \overline{\rho}$ for each leg and returns:

- -1 if $\underline{\rho} > \rho_{max}$ or $\overline{\rho} < \rho_{min}$ for at least one leg
- 0 if $\underline{\rho} < \rho_{min}$ or $\overline{\rho} > \rho_{max}$ for at least one leg, provided that the previous condition is not fulfilled
- 1 if $\underline{\rho} \geq \rho_{min}$ and $\overline{\rho} \leq \rho_{max}$ for all legs

Hence if the $\mathcal{A}$ algorithm returns 1, then we are sure that for any value of the design parameters the leg lengths for the pose is correct: in other words the workspace of any robot geometry whose design parameters have a value in the range will include the pose. A design parameters set $\mathcal{D}$ for which the $\mathcal{A}$ algorithm returns 1 for every pose in the desired set will be called a *valid interval set*. If the $\mathcal{A}$ algorithms return -1, then the pose will never be in the workspace whatever is the value of the design parameters in their ranges. If the $\mathcal{A}$ algorithm returns 0, then we are not able to decide if the range for the design parameters is correct or not.

To determine $\underline{\rho}, \overline{\rho}$ we will rely on a method called *interval analysis* [10]. Basically interval analysis is similar to real number analysis except that the unknowns are defined by ranges and that all the basic mathematical operators must be re-defined. Without going into the details assume that we want to compute bounds on the leg lengths for a given pose when all the design parameters, except R_1, have a fixed value. Then ρ^2 may be written as $\rho^2 = R_1^2 + aR_1 + b$, where a, b are constants. Assume now that $R_1 \in [2, 10]$ and that $a = -2, b = 3$. Clearly under this assumption $R_1^2 \in [4, 100]$, $aR_1 \in [-20, -4]$. Hence ρ^2 is obtained as the sum of the ranges [4,100], [-20,-4], [3,3] and the addition operator in interval analysis indicates that this sum is the range [-13,99]. This means that for any value of $R_1 \in [2, 10]$, then $-13 \leq \rho^2 \leq 99$. We may notice here that interval analysis leads to over-estimated bounds as the real bounds are [3,83]: but

they are numerous tricks in interval analysis that enable to improve the sharpness of the result. Interval analysis may be used for any function that is defined in term of the most classical mathematical functions, such as algebraic terms or the sine and cosine functions that appears in ρ_i. Indeed the square of a leg length may be written as:

$$\begin{aligned}\rho^2 &= R_1^2 + r_1^2 + R_1 r_1 F_1(\alpha,\beta) + R_1 F_2(\alpha,\beta,\rho_d) \\ &\quad + r_1 F_3(\alpha,\beta,\rho_d) + F_4(\alpha,\beta,\rho_d)\end{aligned}$$

if the the z coordinate of the poses is defined with respect to the z coordinate of the platform in its nominal position, or more simply:

$$\begin{aligned}\rho^2 &= R_1^2 + r_1^2 + R_1 r_1 F_1(\alpha,\beta) + R_1 F_2(\alpha,\beta) \\ &\quad + r_1 F_3(\alpha,\beta) + F_4(\alpha,\beta)\end{aligned}$$

if the the z coordinate of the poses is defined with respect to the reference frame. Hence this method may be used to determine the values of $\underline{\rho}, \overline{\rho}$.

3 Algorithm principle

Let us first define a set $\epsilon = \{\epsilon_1, \ldots, \epsilon_5\}$ of five thresholds ϵ_i, one for each of the five design parameters. A design parameters set $\mathcal{D}$ will be said to have a width $w(\mathcal{D})$ lower than ϵ if the width of each 5 ranges in the set is lower than its corresponding value in the set ϵ while $w(\mathcal{D})$ will be larger than ϵ if at least one of its ranges has a width larger than the corresponding ϵ_i.

The purpose of our algorithm is to compute all the design parameters sets $\mathcal{D}$ that are valid and whose width is larger than a given ϵ. In other words we want to compute all the possible ranges for the design parameters that are not too "small" and such that all the desired poses P_i are in the workspace of the corresponding robots. Let also consider a five-dimensional space where each dimension correspond to a design parameter. A point in this space defines a unique robot geometry. A region $\mathcal{R}$ of this space defines all the possible values of the design parameters such that all the desired poses are inside the workspace of the corresponding robots. Our algorithm aims at computing an approximation of $\mathcal{R}$ up to the accuracy ϵ.

Let us consider our 5 design parameters R_1, r_1, α, β and ρ_d. The two angular parameters α, β are naturally bounded. As for ρ_d we can imagine that the choice of the stroke has led to the determination of commercially available actuator whose minimal length is known: hence we have a lower bound for ρ_d. At the same time we may also assume that the total length of the leg cannot exceed too much this minimal value (say cannot be more than twice the minimal value of ρ_d). Thereby we have too an upper bound for ρ_d. As for R_1, r_1 we may impose $R_1 > r_1$ and we will have in general some constraints on the overall size of the robot and hence we may assume that both these design parameters are also bounded.

In conclusion the possible values of the design parameters are initially defined by a design parameters set $\mathcal{D}_1$. A very favourable case will be when $\mathcal{A}(\mathcal{D}_1, P_i) = 1$ for all poses: in that case the workspace of any robot will include the specified poses and we have solved our problem. In a more general case we will use a list $\mathcal{L}$ composed of m design parameters sets and $\mathcal{D}_i$ will denote the i-th element of this list. This list will be initialised with $m = 1$ and $\mathcal{L} = \{\mathcal{D}_1\}$.

We define also a bisection process on a design parameters set $\mathcal{D}_i$: we choose first one range in the set (say $\hat{R_1} = [\underline{R_1}, \overline{R_1}]$) and create the 2 new ranges $\hat{R}_{11} = [\underline{R_1}, (\underline{R_1} + \overline{R_1})/2]$ and $\hat{R}_{12} = [(\underline{R_1} + \overline{R_1})/2, \overline{R_1}]$. We then create 2 new design parameter sets $\mathcal{D}_i^1, \mathcal{D}_i^2$ which have as ranges the same ranges than $\mathcal{D}_i$ except for $\hat{R_1}$ for which $\hat{R_1}(\mathcal{D}_i^1) = \hat{R}_{11}$ and $\hat{R_1}(\mathcal{D}_i^2) = \hat{R}_{12}$. We will not explain how is chosen the bisected variable but interval analysis provides numerous tricks for determining it like, for example, through the smear function [11].

We will also use an index i whose value is the number of the design parameter set that is currently being processed by the algorithm. The algorithm proceeds along the following steps:

1. if $i > m$ return `COMPUTATION COMPLETED`
2. if $w(\mathcal{D}_i) < \epsilon$, then $i = i + 1$, goto step 1
3. if there is a j in [1,n] such that $\mathcal{A}(\mathcal{D}_i, P_j) = -1$, then $i = i + 1$, goto step 1
4. if for all j in [1,n] $\mathcal{A}(\mathcal{D}_i, P_j) = 1$, then `STORE` $\mathcal{D}_i$, $i = i + 1$, goto step 1
5. bisect $\mathcal{D}_i$, store the 2 new design parameters sets at the end of $\mathcal{L}$, $m = m + 2$, $i = i + 1$ and goto step 1

The basic idea of this algorithm is to bisect the parameter sets until we may decide if the current $\mathcal{D}_i$ is valid or not. All the parameters sets that are not valid or are too small are rejected while the valid one are stored and will constitute the output of the algorithm. The algorithm stops when all the parameters sets in $\mathcal{L}$ have been processed by the algorithm. Note that

this algorithm is guaranteed to stop we do not bisect again a variable j that has already a width less than ϵ_j: hence after a finite number of steps either the design parameters set will be such that $w(\mathcal{D}) < \epsilon$ or this set will have been rejected or stored.

4 Failure case and complexity

It must be mentioned that interval analysis may take into account round-off errors. Hence when all the variables of an expression are constants the evaluation of a function through interval analysis may still return an interval that is guaranteed to include the correct value of the function. Thereby even if the range of the design parameters is reduced to a point (which happen only if we set ϵ to 0) it may happen that the $\mathcal{A}$ algorithm will return 0: the algorithm cannot decide if the leg lengths are valid or not. This is the only failure case of the algorithm and can be easily avoided by setting ϵ to a value not equal to 0.

There may be apparently another failure case: the use of the bisection process may lead to a larger number of parameters sets than the memory that was allocated for the storage. In fact it may be shown that a more appropriate storage management than the basic one described in the algorithm allows us to reduce the storage needed to $E(log(w/\mu)/log(2)) + 1$, where w is the largest width of the initial range for the design parameters, μ is the lowest threshold in ϵ and $E(f)$ indicates that we take the integer which is immediately greater than f. For example if $w = 1000$, $\mu = 10^{-5}$ we will need only to be able to store 28 parameters sets. Hence storage is not a problem for this algorithm.

5 Implementation

To implement this algorithm we have used the C++ package `BIAS/Profil`[1] that implement the basic operations on intervals and the high level package `ALIAS`[2] that implement the higher level routines that we need.

A feature of this implementation is that it enable an incremental construction of the approximation of $\mathcal{R}$. Assume that we start a first run of the algorithm with some fixed values in ϵ. When the algorithm encounter a parameter set $\mathcal{D}_j$ such that $w(\mathcal{D}_j) < \epsilon$ and $\mathcal{A}(\mathcal{D}_j) = 0$ for all poses, then the set is written in a file $\mathcal{F}$. After the first run of the algorithm we will get a set of design parameters sets that are valid, that is the approximation of $\mathcal{R}$ up to the accuracy ϵ, and the file $\mathcal{F}$ that contains all the parameters sets for which we have not been able to decide if they belong to $\mathcal{R}$ or not. Now assume that we want to refine this approximation by computing a new approximation of $\mathcal{R}$ with a better accuracy, for example by dividing each threshold in ϵ by 3. In that case there is no need to run the algorithm with the same initial parameters set $\mathcal{D}_1$ that we have been using for the first run. Indeed the only change will be obtained when considering the undecidable parameters sets contained in the file $\mathcal{F}$. Hence instead of initialising $\mathcal{L}$ with $\mathcal{D}_1$ we will initialise it with the parameters sets contained in $\mathcal{F}$. This method enable to construct efficiently a more and more accurate approximation of $\mathcal{R}$.

We may also obtain an index to determine the quality of the approximation of $\mathcal{R}$. Indeed we may compute the volume of the approximation and the total volume of the parameters sets that have been neglected: comparing these values will indicate how close we are from $\mathcal{R}$.

In the implementation we use the gradient of the leg lengths with respect to the design parameters in order to improve the interval evaluation of the leg lengths by using the monotonicity of the functions.

5.1 Computation time

The computation time is dependent upon many factors. A first important factor is if the z coordinates of the poses are defined with respect to the reference frame or with respect to the z of the nominal position. In the later case the leg lengths have a much more complex expression than in the former case, which result in a larger over-estimation when evaluated with interval analysis.

A second important factor is clearly the accuracy as defined by ϵ and the width of the initial range for the design parameters. Indeed the number of design parameters sets that is considered in the algorithm grows exponentially with the inverse of the accuracy and with the width of the initial ranges. For the same reason if one of the design parameters has a fixed value the computation time is reduced drastically. A less important factor is the number of poses in the set $\mathcal{P}$: an index associated to each design parameters set in $\mathcal{L}$ indicates for each pose which leg lengths may not be valid and only the interval evaluation of these lengths will be recomputed.

On a PC laptop the computation time is about a few minutes if the one of the design parameters has a fixed value. On the other hand if all the design parameters are defined by range and the accuracy is about

[1] www.ti3.tu-harburg.de/Software/PROFILEnglisch.html
[2] www.inria.fr/coprin/logiciels/ALIAS/ALIAS.html

0.1% of the width of these ranges the computation is about 12 hours. Note however that by principle this algorithm may be easily configured to be run on a parallel computer or a cluster of computers: in such hardware conditions the computation time may be reduced drastically.

6 Using the algorithm for optimal design

The algorithm is to be used as a first step of a design methodology. It will enable to reduce drastically the search space for the design parameters. Then two approaches are possible:

- we may extend this method to other criteria than the workspace (accuracy or stiffness for example). The corresponding algorithms will enable one to compute other valid regions in the 5-dimensional space. Clearly if there is a solution to the design problem, then the robot geometry must lie within the intersection of all the valid regions (that can be easily computed as we have to calculate only the intersection of sets of hypercubes).

- we may sample the approximation of $\mathcal{R}$: each node in the sampling is a robot geometry and we may apply on it several algorithms that have been developed for analysing the performances of this type of manipulator like workspace evaluation [12] or worst case articular forces [13]. According to the results of these algorithms we may compare all the nodes and retain the best geometry.

7 Examples

In this example we are looking for the robots whose workspace include the four different poses (-5, -5, -5, -10, -10, -10), (-5, -5, 5, -10, -10, -10) (-5, 5, -5, -10, -10, -10), (-5, 5, 5, -10, -10, -10) where the angle are given in degrees. (here the z coordinate of the cube is relative to the z coordinate of the robot in its nominal position). The possible location of C has led us to choose a stroke of 25 units.

The algorithm produces a list of parameters sets which clearly cannot be represented as they are in a five-dimensional space. Figure 2 presents a rough approximation of all the possible values of R_1, r_1, ρ_d with $\hat{R_1} = [20, 50]$, $\hat{r_1} = [2, 40]$, $\hat{\alpha} = [10, 12]$, $\hat{\beta} = [10, 12]$ and $\hat{\rho_d} = [27, 31]$. The number of resulting design parameters sets was 917 for a volume of 0.905 and a neglected volume of 0.462 (a better approximation may easily be obtained but is much more difficult to represent as the number of sets is large). Figures 3 presents

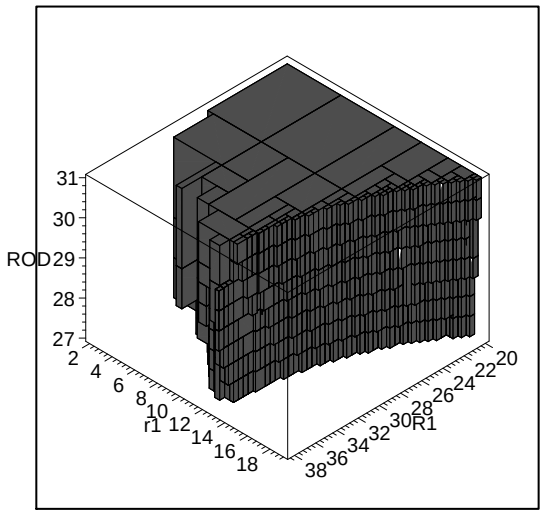

Figure 2: The possible values of R_1, r_1, ρ_d

the result when $r_1 = 20, \rho_d = 28$ while $\hat{R_1} = [20, 50]$, $\hat{\alpha} = [0, 15]$, $\hat{\beta} = [0, 15]$ and the accuracy was defined as 0.2 for R_1, 0.005 for α and 0.01 for β. The result has a volume of 0.0832 while the neglected volume was 0.0163. Figure 4 presents the result when $\alpha = 0.1, \beta = 0.1$ while $\hat{R_1} = [20, 50]$, $\hat{r_1} = [2, 40]$, $\hat{\rho_d} = [27, 31]$ and the accuracy was defined as 0.4 for R_1, 0.2 for r_1 and 0.1 for ρ_d. The result has a volume of 830.955 while the neglected volume was only 58.87.

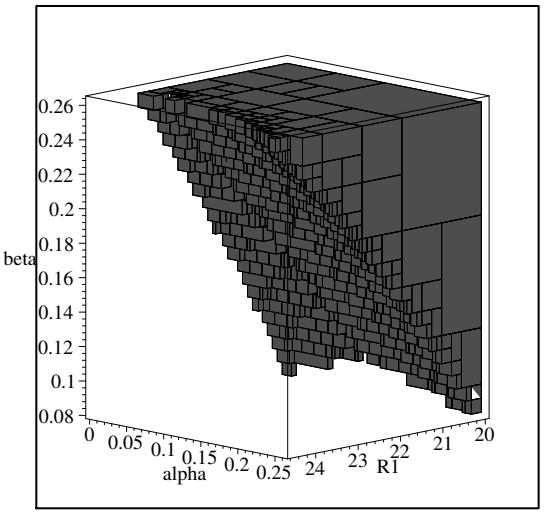

Figure 3: The approximation of $\mathcal{R}$ for fixed values of r_1, ρ_d

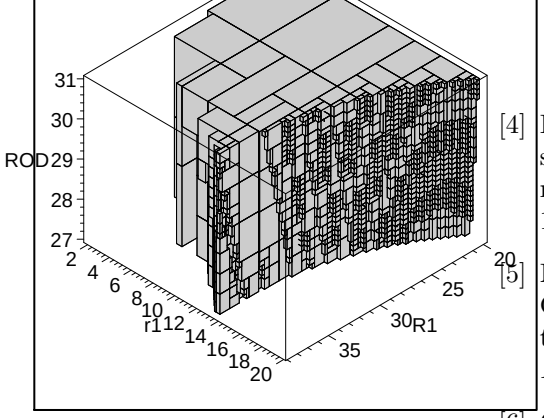

Figure 4: The approximation of $\mathcal{R}$ for fixed values of α, β

8 Conclusion and future works

This work is a first step toward an effective methodology for the optimal design of parallel manipulator whose purpose will be to determine first all the regions of the parameters space which define all the robot geometries satisfying a minimal set of requirements and then search for the optimal design only within this region.

The algorithm has been presented for the Gough platform but may be extended for any architecture having 6 DoF or less (the only change will be to adapt the $\mathcal{A}$ algorithm to the new architecture by changing the function that compute the articular variables).

References

[1] Bhattacharya S., Hatwal H., and Ghosh A. On the optimum design of a Stewart platform type parallel manipulators. *Robotica*, 13(2):133–140, March - April , 1995.

[2] Gosselin C. and Angeles J. The optimum kinematic design of a spherical three-degree-of-freedom parallel manipulator. *J. of Mechanisms, Transmissions and Automation in Design*, 111(2):202–207, 1989.

[3] Zanganeh K.E. and Angeles J. Kinematic isotropy and the optimum design of parallel manipulators. *Int. J. of Robotics Research*, 16(2):185–197, April 1997.

[4] Boudreau R. and Gosselin C.M. The synthesis of planar parallel manipulators with a genetic algorithm. *ASME J. of Mechanical Design*, 121(4):533–537, December 1999.

[5] Murray A.P., Pierrot F., Dauchez P., and McCarthy J.M. A planar quaternion approach to the kinematic synthesis of a parallel manipulator. *Robotica*, 15(4):361–365, July - August , 1997.

[6] Ceccarelli M. and Sorli M. The effects of design parameters on the workspace of a Turin parallel robot. *Int. J. of Robotics Research*, 17(8):886–902, August 1998.

[7] Clavel R. *Conception d'un robot parallèle rapide à 4 degrés de liberté*. PhD thesis, EPFL, Lausanne, 1991. n° 925.

[8] Ji Z. Analysis of design parameters in platform manipulators. *ASME J. of Mechanical Design*, 118(4):526–531, December 1996.

[9] Merlet J-P. Democrat: A DEsign Methodology for the Conception of robots with parallel ArchiTecture. *Robotica*, 15(4):367–373, July - August , 1997.

[10] Moore R.E. *Methods and Applications of Interval Analysis*. SIAM Studies in Applied Mathematics, 1979.

[11] Kearfott R.B. and Manuel N. III. INTBIS, a portable interval Newton/Bisection package. *ACM Trans. on Mathematical Software*, 16(2):152–157, June 1990.

[12] Merlet J-P. Determination of 6D workspaces of Gough-type parallel manipulator and comparison between different geometries. *Int. J. of Robotics Research*, 18(9):902–916, October 1999.

[13] Merlet J-P. Efficient estimation of the extremal articular forces of a parallel manipulator in a translation workspace. In *IEEE Int. Conf. on Robotics and Automation*, pages 1982–1987, Louvain, May, 18-20, 1998.