

A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot.

Jean-Pierre Merlet, David Daney

▶ **To cite this version:**

Jean-Pierre Merlet, David Daney. A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot.. In 2nd Workshop on Computational Kinematics, May 2001, Seoul, South Korea. pp.167-176. hal-00906482

HAL Id: hal-00906482
https://inria.hal.science/hal-00906482

Submitted on 19 Nov 2013

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

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

A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot

J-P. Merlet
INRIA Sophia-Antipolis
France

David Daney
CMW-INRIA
France

Abstract: Determining if there is a singularity within a given workspace of a parallel robot is an important step during the design process of this type of robot. As this singular configuration must be avoided the designer may be interested only in a straight yes/no answer.

We consider in this paper a Gough-type parallel robot and we present algorithms which enable to determine if there is any singularity within a 6D workspace expressed either in term of generalized coordinates (position/orientation of the platform) or articular coordinates (lengths of the legs).

1 Introduction

Parallel robots have been extensively studied this recent years and are now starting to appear as commercial products for various applications, hence the interest for their optimal design. Among other criterion checking singularity is an important part of the design process. In this paper we consider a Gough-type 6 d.o.f. parallel manipulator (figure 1) constituted of a fixed base plate and a mobile plate connected by 6 articulated and extensible links. The pose of the platform may be

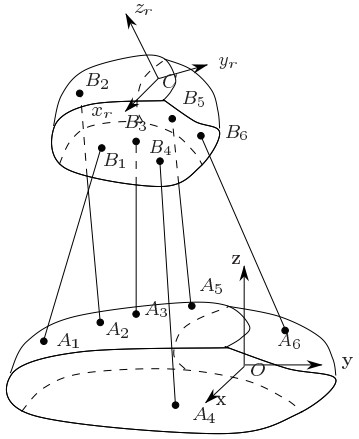


Figure 1: Gough platform

adjusted by changing the length of the six legs. A reference frame (O, x, y, z) is attached to the base and a mobile frame (C, x_r, y_r, z_r) is attached to the moving platform. The leg i is attached to the base with a ball-and-socket joint whose center is A_i, while it is attached to the moving platform with an universal joint whose center is B_i. For a given robot the coordinates of the points are assumed

to be known. Let ρ_i be the leg lengths (the distance between A_i and B_i), $\boldsymbol{X}$ a 6-dimensional vector defining the pose of the end-effector: the three first components of $\boldsymbol{X}$ are the coordinates x_c, y_c, z_c of C in the reference frame, while the three last components are three parameters describing the orientation of the end-effector. In this paper we will use the Euler angles ψ, θ, ϕ but any other representation may be used.

2 Singular configurations

2.1 Inverse jacobian matrix

For a Gough platform it is easy to establish the inverse kinematics as:

$$\boldsymbol{\rho} = \boldsymbol{F}(\boldsymbol{X}) \tag{1}$$

where $\boldsymbol{\rho}$ is the vector of the leg lengths and $\boldsymbol{X}$ the generalized coordinates of the platform. By an appropriate derivation we may obtain the relation between the articular velocities $\dot{\boldsymbol{\rho}}$ and the platform velocities $\dot{\boldsymbol{X}}$ as:

$$\dot{\boldsymbol{\rho}} = J^{-1}\dot{\boldsymbol{X}} \tag{2}$$

where J^{-1} is the inverse jacobian matrix of the robot. A row of this matrix may be written as:

$$J_i^{-1} = \frac{\boldsymbol{A_i B_i}}{\rho_i} \qquad \boldsymbol{CB_i} \times \frac{\boldsymbol{A_i B_i}}{\rho_i} \tag{3}$$

A parallel singular configuration is defined as the configuration where the determinant of J^{-1} vanishes. The importance of determining if there is any singular configuration in the workspace of the robot may be illustrated by considering that at a singular configuration the articular forces may go to infinity, causing a breakdown of the robot.

Note that if we define the *semi inverse jacobian matrix* as:

$$M_i = \boldsymbol{A_i B_i} \qquad \boldsymbol{CB_i} \times \boldsymbol{A_i B_i} \tag{4}$$

we may see that:

$$|J^{-1}| = \frac{|M|}{\prod_{i=1}^{i=6} \rho_i} \tag{5}$$

Therefore the singular configuration may also be defined as the configuration for which $|M| = 0$.

Finding all the singular configurations of a given robot is a difficult task: indeed, although the inverse jacobian matrix is known, expansion of its determinant leads to an huge expression [2] which is difficult to use. Another approach is based on Grassmann line geometry and has been successful to determine all the possible relations on the pose parameters leading to a singularity [4].

But from the design view point we are more interested in the following question: is there a singularity(s) in a given workspace of a given robot?. The designer may be interested by a binary answer (yes or no) and our algorithms will provide this answer. To the best of our knowledge this problem has not yet been addressed in the literature for 6 d.o.f. robot. Related works deal with conditioning index (like the condition number of the inverse jacobian matrix estimated over the whole workspace [1]). But this index is usually estimated by a discrete method over the whole workspace and henceforth may fail to locate singularities.

3 Existence of a singular configuration

We consider here a workspace W defined as a closed region in the 6-dimensional generalized coordinates space. We consider a point X_1 taken at random in W. We compute $|M|$ at point X_1 and we will assume, without loss of generality that $|M(X_1)|$ is greater than zero. Let us assume that we are able to find a point X_2 (or a set of points), in the same component of the workspace than X_1, such that $|M(X_2)| < 0$. As $|M|$ is a real valued continuous and differentiable function, then any path from X_1 to X_2 will have a point X_s for which $|M(X_s)| = 0$, which means that X_s is a singular configuration of the robot. Hence as soon as a point X_2 has been determined we know that there is a singular configuration in the workspace of the robot.

The purpose of the following sections is to present algorithms which are able to determine a pair of points X_1, X_2, if any exist. If such pair is found, then we have proven the existence of at least a singular configuration within the workspace and if no such pair has been found, then we will have proven that the workspace is singularity-free.

4 Workspace and extended box

4.1 Extended box

We define an *extended box* (or EB for short) as a pair of element: a cartesian box, which represent the possible locations of the end-effector, and a set of three ranges, one for each of the rotation angles. An EB is therefore composed of a *location part* (the box) and an *orientation part* and defines a 6D workspace for the robot.

4.2 Bisection of an extended box

In the sequel we will use an operator for an EB called *bisection*. This operator takes as input an EB and outputs up to 64 new EBs whose union is the initial EB. They are obtained by first splitting in half some of the 6 ranges $I_1, \ldots, I_6$ which define the input EB. Thus if $I_i = [a_i, b_i]$ we get two new ranges I_{i1}, I_{i2}, namely

$$I_{i1} = [a_i, (a_i + b_i)/2] \quad I_{i2} = [(a_i + b_i)/2, b_i]$$

The new EBs are then obtained by considering all the possible combinations of the bisected ranges and non bisected ranges For example if the input EB is defined by the following ranges:

$$[-1, 1], [-1, 1], [40, 50], [-1, 1], [-1, 1], [-1, 1]$$

and if we bisect only the two first ranges, then the output of the bisection will be the 4 EBs defined by:

$$\{[-1, 0], [-1, 0], [40, 50], [-1, 1], [-1, 1], [-1, 1]\} \quad \{[-1, 0], [0, 1], [40, 50], [-1, 1], [-1, 1], [-1, 1]\}$$
$$\{[0, 1], [-1, 0], [40, 50], [-1, 1], [-1, 1], [-1, 1]\} \quad \{[0, 1], [0, 1], [40, 50], [-1, 1], [-1, 1], [-1, 1]\}$$

4.3 Singularity detection in a workspace

Consider an EB $\mathcal{B}$ and let assume that we have an algorithm $\mathbf{A}(\mathcal{B})$ which is able to determine first a lower and an upper bound of $|M|$ for any pose within $\mathcal{B}$ and will then return: 1, if the lower bound is positive, -1, if the upper bound is negative and 0, if the lower bound is negative and the upper

bound is positive. The bounds provided by the **A** algorithm may be overestimated but should be exact if the EB is reduced to one pose. The purpose of this section is to show that then we are able to determine if a singularity occurs within any type of workspace.

4.3.1 Singularity in a generalized coordinates workspace

Let assume that we want to check the presence of singularity in a workspace W defined by a 3D volume V describing the possible location of the center of the end-effector and three ranges I_ψ, I_θ, I_ϕ which describe the possible values of the orientation angles. Clearly V may be approximated by a set of EB.

Let us assume now that we are able to design a test algorithm $\mathbf{T}(\mathcal{B})$ which enable to determine if for the EB $\mathcal{B}$:

1. the location part of $\mathcal{B}$ is fully inside V
2. the location part of $\mathcal{B}$ is fully outside V
3. the location part of $\mathcal{B}$ is partially inside V

The algorithm **T** will return an integer which may 1, 2 or 3 according to the status of the location part with respect to V.

Using the algorithms **A** and **T** we are able to design an algorithm which enable to determine if a singularity occur within W. The algorithm start by computing a list S=$\{\mathcal{B}_0, \mathcal{B}_1, \ldots \mathcal{B}_{n-1}\}$ of n EBs such that the union of the location part of the EBs in the list is an approximation of the volume V, strictly including V, while the orientation part of each EB is simply defined as the range I_ψ, I_θ, I_ϕ. We will then use two flags f_-, f_+ which will be initialized to 0 and will be set to 1 as soon as an EB for which the upper (lower) bound of $|M|$ is negative (positive) is encountered during the algorithm. We may now describe the different steps followed by the algorithm at iteration k, the algorithm starting at iteration 0:

1. if $k > n-1$ return **NO SINGULARITY**
2. if $\mathbf{T}(\mathcal{B}_k) = 2$ then $k = k+1$ and reiterate
3. if $\mathbf{A}(\mathcal{B}_k) = 1$ and $\mathbf{T}(\mathcal{B}_k) = 1$ then set f_+ to 1
   - if $f_- = 1$ then return **SINGULARITY**
   - otherwise $k = k+1$ and reiterate
4. if $\mathbf{A}(\mathcal{B}_k) = -1$ and $\mathbf{T}(\mathcal{B}_k) = 1$ then set f_- to 1
   - if $f_+ = 1$ then return **SINGULARITY**
   - otherwise $k = k+1$ and reiterate
5. if $\mathbf{T}(\mathcal{B}_k) = 3$ or $\mathbf{A}(\mathcal{B}_k) = 0$ bisect $\mathcal{B}_k$ and put the new EBs at the end of the list S whose number of elements is updated, $k = k+1$ and reiterate

Basically this algorithm just consider each EB of the list sequentially. Flags are set as soon as an EB included in W is such that $|M|$ has a constant sign. As soon we have encountered an EB with positive $|M|$ and an EB with negative $|M|$ we may state that a singularity occurs within W. During the iteration we will encounter EBs which are only partially inside W, or fully inside W but

with bounds on $|M|$ which have opposite sign. In that case we just bisect the EB and the resulting EBs are added to the list.

The algorithm stop either if a singularity has been detected or when all the elements of S have been considered, in which case there is no singularity in W.

4.3.2 Singularity in an articular workspace

Let us assume now that the workspace is not defined in term of generalized coordinates but in term of articular coordinates i.e. as all the poses X for which the leg lengths satisfy:

$$\rho^i_{min} \leq \rho^i(X) \leq \rho^i_{max}$$

in which $\rho^i(X)$ denote the length of leg i at pose X and $\rho^i_{min}, \rho^i_{max}$ the minimal and maximal lengths of leg i. In other words we want to detect if a singularity occur within the *reachable workspace* of the robot.

Basically we will use exactly the same algorithm than in the previous section, with only a few modifications.

The first modification will be related to the algorithm **T**: we will use here an algorithm which enable to determine first what will be the extremal values of the leg lengths for any pose within an EB [3].

Then the design of **T** is quite straightforward. Indeed, let $\rho^i_m(B), \rho^i_M(B)$ denote the minimal and maximal value of the length of leg i for the EB $\mathcal{B}$, as computed by the algorithm. Then algorithm **T** will be

- if for all legs $\rho^i_{min} \leq \rho^i_m(B), \rho^i_M(B) \leq \rho^i_{max}$ then return 1
- if for at least one leg we have $\rho^i_m(B) > \rho^i_{max}$ or $\rho^i_M(B) < \rho^i_{min}$ then return 2
- otherwise return 3

Note that it is not necessary to compute the exact values of $\rho^i_m(B), \rho^i_M(B)$, except in the case where $\mathcal{B}$ is reduced to a pose. Only lower and upper bounds on the minimal and maximal values are necessary. Indeed, the only consequence of an error on the estimation of these extremal values is that the algorithm **T** may return a status 3 for an EB whose real status is 1. In the algorithm an EB with status 3 will be bisected until the status of the new EBs is either 1 or 2. If the real status of the EB is 1, then after a few bisection all the EBs will have status 1.

The second modification of the previous algorithm is to determine the initial EBs of the list S. This problem may be solved by finding an EB whose location part is a bounding box of the reachable workspace. Indeed for the orientation part as we deal with angle we have natural bounds on the range with the interval $[0, 2\pi]$. But C is at fixed distance from the B_i which are in turn constrained to be within a maximum distance from the A_i: consequently finding such bounding box is an easy task.

With such initialization of S and by using the new **T** algorithm we may now use the previously described singularity detection algorithm to find if a singularity occur within an articular workspace.

More precisely if the previous algorithm does not detect a singularity then we may state that there is no singularity in the articular workspace. On the contrary let us assume that we have detected 2 EB in the articular workspace for which $|M| > 0$ and $|M| < 0$. A singularity will occur in the articular workspace only if these two regions belong to the same connected component of the articular workspace. Currently, to the best of my knowledge, there is no known method that may ensure that the 2 EBs belong to the same connected component.

5 Interval analysis

5.1 Principle

As we have seen in the previous section detecting singular configurations amounts to be able to find two EBs such that $|M|$ for any pose in the EBs has a constant sign, this sign being opposite for the EBs. To ensure that $|M|$ has a constant sign over an EB we may compute the minimal and maximal value of $|M|$ for all the poses included in the EB. It must be noted that it is not necessary to compute exactly these extremal values: lower bound for the minimal value and upper bound for the maximal value will be sufficient.

Interval analysis [5] is a method that is able to deal with the problem of determining bounds on a function. Interval analysis is similar to real analysis except that real variables are replaced by intervals and that specific rules are used for each basic arithmetic operations.

More generally if $X_i = [\underline{x_i}, \overline{x_i}]$, with $\underline{x_i} \leq \overline{x_i}$, denotes an interval it is possible to define all the arithmetic operators (and more complex functions) for intervals. For example the "+" operator for intervals will be defined as:

$$X_1 + X_2 = [\underline{x_1} + \underline{x_2}, \overline{x_1} + \overline{x_2}]$$

Consider now a very simple example of interval analysis: let the function $x^2 + x$ for which we want to find the extremal values when x lie in the range [-1,0]. We consider each monomial of this function and compute its extremum, which lead to $x^2 \in [0, 1]$, $x \in [-1, 0]$. Then the upper bound of the sum is computed as the sum of the upper bound of each interval while the minimum is obtained as the sum of the lower bound.

$$x^2 + x = [0, 1] + [-1, 0] = [-1, 1]$$

An important point is that an analytical formula may be written in various forms and the corresponding interval evaluation will depend upon the form (in term of interval analysis we say that there are different *evaluation functions*). For example we may have used the Horner (or "nested") form of the previous function and get the following result:

$$x^2 + x = x(x + 1) = [-1, 0] \times [0, 1] = [-1, 0]$$

We notice here that the Horner form leads to a better estimation of the maximum (the real extremum being [-1/4,0]).

A good point of interval analysis is that it can take into account round-off errors: result obtained through interval analysis may be guaranteed: the interval which is the result of operations on a set of interval is guaranteed to include the exact extremum values as computed using exact arithmetics.

Numerous packages of interval analysis are available. In our implementation we are using the `BIAS/PROFIL` package[1].

6 Detecting singularity in an extended box

6.1 Finding bounds on the determinant for an extended box

6.1.1 A formal-numerical approach

As all the unknowns in $|M|$ are ranges we may apply interval analysis to obtain a lower bound of the minimal value of $|M|$ and an upper bound on the maximal value of $|M|$, as soon as an analytical

[1] http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html

formulation of $|M|$ is known. The problem is now to find this analytical formulation, under the constraint that we want to implement a general-purpose program that is able to deal with any robot geometry, being understood that computer algebra program like MAPLE have difficulties to compute a generic form of $|M|$, although this is possible if the coordinates of the A_i, B_i points have numerical values.

We propose a first approach which mix formal and numerical computation. When running our algorithm we will first start a MAPLE program that will first read in a file the coordinates of A_i, B_i points and then compute the semi-inverse jacobian and its determinant (which is at this step only a function of $x, y, z, \psi, \theta, \phi$). The MAPLE program will then decompose this determinant as a sum of terms of the form

$$F_j(x, y, z, \psi, \theta, \phi) = C_j x^i y^J z^k \cos^l(\psi) \sin^m(\psi) \cos^n(\theta) \sin^o(\theta) \cos^p(\phi) \sin^q(\phi)$$

where C_j is a numerical constant. The MAPLE program will then write in a result file the number of terms F_j and a description of each term, namely the $i, J, k, l, m, n, o, p, q, C_j$ coefficients. The main program will then read the result file and fill an appropriate data structure which enable to store each F_j. Using this structure the program is then able to compute an interval evaluation of $|M|$ for any range of the pose parameters. Note that this computation has to be done only once for a given robot and afterwards any workspace can be tested.

Note that in this method the computation of the coefficients has to be done quite carefully. Indeed they result from a large number of calculation using the coordinates of the attachment points. Numerical errors in these calculations may lead to a bad estimation of the determinant of M. To avoid this problem we transform the coordinates of the attachment points into their exact rational form (as only a limited number of digits are significant for the coordinates) and then compute the determinant which will be then exact (and the coefficients C_j will be integers).

Still this process may lead to numerical problems as for some robot geometries the coefficients C_j may be larger than the maximal integer that has a representation in a computer. To solve this problem we use in our implementation two possible representations for the coefficients C_j. If the computed coefficient is lower than the largest available integers, then C_j is represented by this integer, otherwise C_j is represented by an interval which is guaranteed to include the real value of the coefficient. This later representation has the major drawback that for very large coefficients C_j, the width of the interval used to represent a coefficient may be also large. As a consequence we may end up in a situation in which, even for a fixed pose, the evaluation interval of $|M|$ may be a large interval which will include 0, in which case all of the algorithms presented in section 4.3 will fail.

Clearly the algorithm which determine the presence of singularity in an EB is a key point of all the algorithms which deal with the different types of workspace. A crucial point is the determination of the lower and upper bounds of $|M|$. Clearly the closer these bounds are to the real minimum and maximum values, the faster will be the algorithm. We have therefore to deal with a major problem of interval analysis which is that this method may lead to largely over-estimated bounds.

There are numerous methods to improve the interval evaluation of $|M|$. Let us mention two of them:

- *use the monotonicity*: using MAPLE we are able to compute the derivatives of $|M|$. Interval evaluation of these derivatives may lead to one or more derivative intervals having bounds with the same sign. Hence $|M|$ will be monotonous with respect to these variables. Consequently the evaluation of $|M|$ will be done using fixed value for this variable, thereby improving the evaluation. Note that this is a recursive process: having a variable being a number instead of an interval may change the interval evaluation of the derivatives with respect to the other variables.

- *use the 3B method*: let assume than during the process it has been found that for one EB $|M|$ is always positive (hence we are looking for an EB for which $|M|$ will be negative). Consider now one of the pose parameters, say x_c, which has currently for range $[x_1, x_2]$ and a small positive number ϵ. We substitute the range for x_c by the range $[x_1, x_1 + \epsilon]$ and compute the interval evaluation of $|M|$: if this evaluation lead to an always positive value for $|M|$, then a negative value for $|M|$ may be obtained only for x_c in the range $[x_1 + \epsilon, x_2]$. We repeat first the process with this new interval, until the evaluation of $|M|$ is no more strictly positive, then for the right side of the interval for x_c and, finally, for all the other variables. This enable to reduce the width of the ranges for the unknowns, thereby improving the interval evaluation.

6.1.2 A numerical approach

Another approach is to consider that we have to deal with a specific type of constrained optimization problem: we want to determine the minimum and maximum values of $|M|$ being given bounds on the unknowns (and for the articular workspace with the additional constraints that the leg lengths should be within some limits) and exit from the optimization process as soon as it is shown that the values will have opposite sign. At the same time we are looking for an optimization algorithm that guarantee to provide global extremum as this is a necessary to ensure the completeness of the singularity detection. For this purpose we have used the interval-based C++ package `ALIAS`[2] that provide such type of optimization procedure (the extremum are computed up to a given accuracy). There are five main advantages in using `ALIAS` for our problem:

- it is able to deal with almost any mathematical functions
- the function to be optimized may include terms defined as determinant of matrices without having to give an analytical formulation of the determinant (only the component of the matrices have to be defined)
- it is fully interfaced with MAPLE: in our case we have just to define the M matrix and call a specific MAPLE procedure that will automatically generate the C++ code for the optimization problem at hand
- it enable parallel computation on a set of computers based on the `pvm` package[3]. Clearly the singularity detection algorithm has a structure that is highly favorable for parallel computation as every EB in the list S can be processed independently
- if there is no singularity within the workspace we will get the extremum of $|M|$ which may be considered as a measure of the conditioning of the robot

The algorithm for optimization procedure is basically identical to the algorithm described in section 4.3 except that the evaluation of the determinant of M will be obtained by a minor expansion taking into account that these coefficients are ranges. Furthermore a local optimizer will try to determine the local optimum within the current EB using a steepest descent method.

Using an interval evaluation of the components of the matrix for computing $|M|$ has the advantage compared to the previous method that we have no more any numerical problem, while having the drawback that the interval evaluation of $|M|$ will be largely more overestimated than before as we are no more taking into account the simplification that may occur formally during the computation of the determinant. Our experiments has shown that indeed this method was largely slower but very robust.

[2] `www.inria.fr/saga/logiciels/ALIAS/ALIAS.html`
[3] http://www.netlib.org/pvm3/book/pvm-book.html

7 Dealing with uncertainties in the location of the joints

Up to now we have assumed that the geometry of the robot was perfectly known. In practice however manufacturing tolerances lead to uncertainties in the locations of the joints. We may assume that the coordinates of the joints may be represented by small ranges instead of floating point numbers, the purpose being to determine if there is a singularity in the workspace for any real robot whose joint coordinates belong to the ranges. We may still theoretically proceed with the calculation of the determinant of the semi inverse jacobian matrix, the coefficients C_j being now ranges and consequently we may apply the singularity detection algorithm. But a problem may arise: if the width of the ranges C_j are too large we may be not able to determine the sign of $|M|$ even at a fixed pose, in which case the algorithm will fail.

8 Examples and computation time

The above algorithms have been implemented in C++ on a SUN Ultra 10 workstation under Unix-Solaris and on a PC laptop Dell Latitude CP 300 (300 Mhz) under linux. The computation time provided for the example have been obtained on these computers. They do not include the computation time of the MAPLE session. All the angle values are given in degree.

8.1 The considered robots

The coordinates of A and B of robot 1 and 3 are presented in the table 1. For robot 1 the minimal

	x_A	y_A	z_A	x_B	y_B	z_B
1	-9	9	0	-3	7	0
2	9	9	0	3	7	0
3	12	-3	0	7	-1	0
4	3	-13	0	4	-6	0
5	-3	-13	0	-4	-6	0
6	-12	-3	0	-7	-1	0

	x_A	y_A	z_A	x_B	y_B	z_B
1	-2539	1778	0	-657	-239	-100
2	2539	1778	500	657	-239	100
3	2809	1310	0	121	689	-100
4	270	-3088	500	-536	-449	100
5	-270	-3088	0	536	-449	-100
6	-2809	1310	500	-121	689	100

Table 1: Coordinates of the A, B points for robot 1 and 3

and maximal lengths of the legs are 55 and 60, while for robot 3 the minimal and maximal length for legs 1, 3, 5 are 3760, 4390, while for legs 2, 4, 6 they are 3545,4175.

8.2 Singularity in an extended box

The considered EB is defined by the following range:

x	y	z	ψ	θ	ϕ
[-15,15]	[-15,15],	[45,50]	[-15,15]	[-15,15]	[-15,15]

For this extended box the absence of singularity is verified in 51.41s on the SUN workstation and in 24.61s on the PC.

Singularity in an extended box have been also studied for robot 3. In the EB defined by [-200,200], [-200,200], [2800,3200], [-20,20], [-20,20], [-20,20] no singularity are detected in 19m5s on the SUN and 29mn11s on the PC.

8.3 Singularity in an articular workspace

For robot 1 we have considered the articular workspace obtained when the angles ψ and ϕ are equal to 0 while the range for the angle θ is [-40,40]. The absence of singularity is verified in 6.54s on the SUN workstation and 3.71s on the PC.

For robot 3 we have investigated the presence of singularity in the articular workspace for which we have restricted the z coordinates of the end-effector to be greater than 2000, and the Euler's angles to be in the range [-20,20]. The initial EB in S has the following ranges for the x, y, z coordinates:

$$x \in [-1878.85, 1644.15] \quad y \in [-2674.69, 1364.79] \quad z \in [2000, 5096.24]$$

After examining 9147 EBs no singularity is detected in this articular workspace: the computation time on the SUN workstation is 1h43mn and 2h19mn on a PC.

9 Conclusion

The algorithms proposed in this paper enable to solve a very important problem in the design process of a parallel robot. As may be seen in the benches the computation time is in general quite acceptable although we believe that large gain may still be obtained.

Both methods are interfaced through MAPLE: a direct consequence is that although the singularity detection algorithm has been presented for the Gough platform it may be used for any mechanical architecture just by providing a MAPLE procedure that compute the inverse jacobian matrix.

References

[1] 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.

[2] Mayer St-Onge B. and Gosselin C. Singularity analysis and representation of spatial six-dof parallel manipulators. In J. Lenarčič V. Parenti-Castelli, editor, *Recent Advances in Robot Kinematics*, pages 389–398. Kluwer, 1996.

[3] Merlet J-P. Finding the extrema of the leg lengths of a Gough-type parallel robot when the platform is moving in a given 6D workspace. In *10th World Congress on the Theory of Machines and Mechanisms*, pages 86–91, Oulu, June, 20-24, 1999.

[4] Merlet J-P. Singular configurations of parallel manipulators and Grassmann geometry. *Int. J. of Robotics Research*, 8(5):45–56, October 1989.

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