A Formal-Numerical Approach for Robust In-Workspace Singularity Detection

J.-P. Merlet, *Member, IEEE*

***Abstract*—Singularity is a major problem for parallel robots as in these configurations the robot cannot be controlled, and there may be infinite forces/torques in its joints, possibly leading to its breakdown. Hence, such a configuration must usually be avoided, and certifying the absence of singularity within a prescribed workspace or on a given trajectory is essential for a practical use of this type of robots. Singularity conditions are usually quite complex, and therefore a purely analytical analysis is difficult. We present here an algorithm that combines formal and numerical calculations for detecting singularity or closeness to a singularity within an arbitrary workspace or trajectory. This algorithm has the very important advantage of being able to deal with any robot mechanical structure and to manage uncertainties in the robot control and in the robot modeling.**

***Index Terms*—Interval analysis, inverse Jacobian, parallel robot, singularity.**

I. Introduction

PARALLEL robots have been extensively studied these last few years and are now used as commercial products for a large variety of applications (e.g., packaging, telescopes, machine-tool, and fine positioning). However, their study is still young compared with their serial counterparts. Singularity analysis is one of the many problems that have to be investigated, and we recall briefly this topic.

A first approach to introduce singularity is to consider the mechanical equilibrium of parallel robots. Let $\mathcal{F}$ be a six-dimensional wrench applied on the end-effector that should be counterbalanced by the internal forces/torques in the joints of the structure. Static analysis allows to establish a linear relation between the joint forces/torques τ, and the wrench $\mathcal{F}$ as

$$\mathcal{F} = J^{-T}(X)\tau \tag{1}$$

where J^{-T} is the transpose of the inverse Jacobian matrix J^{-1}, which depends on the pose X of the robot. Given $\mathcal{F}$, we may calculate the joint forces τ by solving the linear system (1). Each joint force will be obtained as a ratio whose denominator is the determinant of J^{-T}. Clearly, the poses in which this determinant cancels are problematic: as soon as the robot gets closer and closer to such a pose, the determinant will get closer and closer to 0, thereby leading to larger and larger joint forces. Such large forces have led to the breakdown of some academic and industrial prototypes, although this has not been largely advertise. A *singular pose (or singularity)* is thus defined as a pose in which the matrix J^{-T} is singular.

Note that (1) is valid for any robot, whatever its degrees of freedom (DOF). Indeed, for robots having less than 6 DOF, it is usual to write a reduced linear system, with matrix J_r^{-T}, which involves only the actuated joint forces/torques and the wrench associated to the controlled DOF of the robot. Still, the end-effector is a rigid body on which may be applied an arbitrary wrench; hence, (1) has to be used, although it is necessary to have more elements in τ. Furthermore, it has been shown that J^{-1} may be singular, while J_r^{-1} is not. For example, this is the case of the 3-UPU robot, as shown by Bonev and Zlatanov [1] and later on by Di Gregorio [2].

Hence, regardless of the robot, a singularity analysis has to consider first the 6×6 J^{-1} matrix, which we will call the *full* inverse Jacobian, and not the reduced inverse Jacobian matrix. The reduced matrix will be considered if and only if it may be shown that the singularity of J^{-1} corresponds exactly to the singularity of J_r^{-1}.

Singularity may also be introduced by analyzing the velocities relations. Let W be the 6-D twist of the end-effector, and let $\dot{\Theta}$ be a joint velocities vector, which is constituted of the actuated joint velocities, possibly augmented by $\mathbf{0}$ if the robot has less than 6 DOF. The end-effector and joint velocities are linearly related by

$$\dot{\Theta} = J^{-1}(X)W. \tag{2}$$

In a singularity, there are nonzero twists that are solutions of the system (2) for a zero-joint velocities vector. The end-effector exhibits infinitesimal motion although the actuators are locked: the robot can no longer be controlled. The loss of control and infinite joint force clearly justify an in-depth singularity analysis. This is not an easy task, although to the best of the author's knowledge it is possible to establish the matrix J^{-1} analytically for all parallel robots (in this paper, we will assume that this matrix is available). However, the complexity of its determinant, whose cancellation characterizes a singularity, increases very quickly with the number of DOF of the robot, although a compact form may sometime be obtained by using Grassmann–Cayley algebra [3]–[5]. Still, for a given robot, symbolic computation may allow is to establish a closed form of the determinant as a function of the pose parameters, even for 6-DOF robots [6]. However, if the geometrical parameters of the robot are added as unknowns, it may become impossible to get a closed form of the determinant.

Manuscript received June 7, 2006; revised November 23, 2006. This paper was recommended for publication by Associate Editor F. Thomas and Editor F. Park upon evaluation of the reviewers' comments. This work was supported in part by the European NEST STREP Project No. 015653 ARES.

The author is with INRIA, 06902 Sophia-Antipolis, France (e-mail: Jean-Pierre.Merlet@sophia.inria.fr).

Digital Object Identifier 10.1109/TRO.2007.898981

To analyze the structure of singularity, it is possible to use Grassmann line geometry that provides geometrical conditions for a linear dependency between the rows of the inverse Jacobian and allows us to establish an exhaustive classification of singularities [7]. However, the geometrical approach does not provide direct clues as to the location of the singularity. Hence, it may not be the most appropriate for determining if there is a singularity for a robot with n DOF in a $m \leq n$-dimensional motion variety (if $m = 1$, the objective is to check a trajectory, while if $n = m$ part of the robot workspace is checked). As mentioned earlier, singularity detection is critical for a practical use of parallel robots. Still, to the best of the author's knowledge, a singularity detection algorithm has never been proposed, although schemes for avoiding badly conditioned poses may be found [8]–[14]. Let us define the robot *useful workspace* for a given task as the part of its workspace in which the end-effector will be the most commonly located. We may accept a relatively large offline computation time to check the useful workspace as it will not be necessary later on to verify online that the robot trajectories included in the useful workspace are singularity-free. However, it may happen that the robot may have to execute from time to time motion varieties such as trajectories that may lie (at least partly) outside its useful workspace. Hence, checking the singularity on $n < m$-dimensional motion varieties is also of interest, but, in that case, the computation time of the singularity detection algorithm should be as low as possible. Uncertainties occur at two levels:

- for $m < n$ motion varieties, there will be unavoidable control errors;
- in all cases, there will be differences between the real geometry of the robot and its theoretical model.

However, the task at hand may impose a safe verification of the absence of singularity in spite of these uncertainties (e.g., for medical applications or for safety reasons in large systems). The purpose of this paper is to provide a guaranteed singularity detection scheme, even in the presence of numerical round-off errors and bounded uncertainties for the geometrical and control parameters. Note that the singularity detection scheme presented in this paper may be modified to locate singularities. Due to space constraints, these modifications will not be exposed here, but the principle of a singularity localization scheme for a trajectory may be found in [15].

II. Singularity Detection

We consider an n-DOF robot whose pose is defined by n parameters grouped in the vector $\boldsymbol{X} = \{x_1, \ldots, x_n\}$. This robot is constrained to move within a $m \leq n$-dimensional variety which we will call the *motion variety* that is defined by the end-user. We will assume that this variety is parametric, i.e., each pose belonging to the variety is such that

$$x_i = f_i(t_1, \ldots, t_m) \qquad \forall\, i \in [1, n] \tag{3}$$

where $\boldsymbol{\mathcal{T}} = \{t_1, \ldots, t_m\}$ is a set of parameters allowing to describe the variety, which each t_j being constrained to lie in the range [0, 1]. Examples of such variety are:

TABLE I
Algorithm 1: Singularity Detection Scheme: $|\boldsymbol{J^{-1}}| > \alpha$?

```
1)  $P = 0$
2)  $k = 0$
3)  $r_k$=1
4)  while ($k \leq r_k$) do
5)    compute $\mathcal{M}(\mathcal{B}_k) = [\underline{m_k}, \overline{m_k}]$
6)    if ($\underline{m_k} > \alpha$) then
7)      $r_{k+1} = r_k$
8)      $k = k + 1$
9)    else
10)     if ($\overline{m_k} \leq \alpha$) then
11)       return SINGULARITY
12)     end if
13)   end if
14)   if ($\underline{m_k} < \alpha$ and $\overline{m_k} > \alpha$) then
15)     if $w(\mathcal{B}_k) = 0$ then
16)       $P = 1$
17)       $r_{k+1} = r_k$
18)       $k = k + 1$
19)     else
20)       bisect $\mathcal{B}_k$ into $\mathcal{B}_k^1$, $\mathcal{B}_k^2$
21)       store $\mathcal{B}_k^1$ as $\mathcal{B}_k$
22)       if ($k = r_k$ or $w(\mathcal{B}_k^2) > w(\mathcal{B}_{r_k})$) then
23)         store $\mathcal{B}_k^2$ as $\mathcal{B}_{r_k+1}$
24)       else
25)         forall $n$ in $[r_k, k+1]$ do
26)           $\mathcal{B}_{n+1} = \mathcal{B}_n$
27)         end forall
28)       end if
29)     end if
30)   end if
31) end while
32) if ($P = 0$) then
33)   return NO SINGULARITY
34) else
35)   return POSSIBLE PROBLEM
36) end if
```

- a box workspace defined by $x_i = x_i^0 + t_i(x_i^1 - x_i^0) \quad \forall\, i \in [1, n]$: the pose parameter x_i has to lie in the range $[x_i^0, x_i^1]$, where x_i^0, x_i^1 are known constants;
- a trajectory defined by $x_i = f_i(T)$, where the parameter T is the time and f_i is an arbitrary analytical function.

It will be assumed that the motion variety has only a single component and is fully included in the configuration space of the robot (i.e., the set of poses that can be reached by the robot). For a given motion variety, this assumption may be verified by using the algorithms presented in [16] and [17].

The purpose of our detection scheme is to determine if there is a pose in the motion variety such that the absolute value of $|\boldsymbol{J^{-1}}|$ is lower than or equal to a given threshold $\alpha \geq 0$. If α is set to 0, we will be able to determine if a singularity occurs within the motion variety. If $\alpha > 0$, its value should be chosen in such way that a pose with the absolute value of $|\boldsymbol{J^{-1}}|$ lower than α is unsafe from a control viewpoint. In other words, the determinant is chosen as an index to measure the closeness to a singularity as it is not possible to define a proper mathematical "distance" between a pose and a singularity [18]–[20].

For a given motion variety, we assume that we are able to select a value for each parameter t_j so that we obtain a pose $\boldsymbol{X_1}$ belonging to the variety for which we may assert that $|\boldsymbol{J^{-1}}| < -\alpha$ or $|\boldsymbol{J^{-1}}| > \alpha$. This assumption is not strictly necessary but will simplify the explanation of the algorithm. Without lack of generality, we will assume in the remainder of this paper that

$|\boldsymbol{J^{-1}}(\boldsymbol{X_1})| > \alpha$. If $\alpha = 0$, a singularity will occur within the variety if we are able to find a pose $\boldsymbol{X_2}$ at which $|\boldsymbol{J^{-1}}(\boldsymbol{X_2})| \leq 0$. If $\alpha > 0$ and if we are able to find a pose $\boldsymbol{X_2}$ at which $|\boldsymbol{J^{-1}}(\boldsymbol{X_2})| \leq \alpha$, we will estimate that the robot will be too close to a singularity for a safe control. Our singularity detection algorithm is based on the determination of such a pose $\boldsymbol{X_2}$.

This determination may be seen as a special optimization problem, for which we have simply to prove that the minimal value of the determinant will be lower than or equal to α. Usually, it will not be possible to solve analytically this optimization problem, and numerical optimization methods cannot guarantee that we will find a pose $\boldsymbol{X_2}$, even if one such pose exists in the motion variety. Furthermore it appears that optimization procedures have some difficulties minimizing or maximizing the determinant over a given motion variety: for instance, `Maple` fails to determine the minimum of the determinant of a Gough platform over a simple 6-D workspace even if there is no uncertainty on the robot geometry.

We will use a method that allows us to evaluate bounds for the minimal and maximal values of $|\boldsymbol{J^{-1}}|$ when the parameters t_j are constrained to lie in some ranges included in [0, 1], i.e., we will determine two values $\underline{m}, \overline{m}$ such that $\underline{m} \leq |\boldsymbol{J^{-1}}| \leq \overline{m}$ for any instance of the t_j in their ranges. These bounds will be numerically safe but may overestimate the minimum and maximum of $|\boldsymbol{J^{-1}}|$. The calculation of these bounds is obtained with interval analysis, which is a method that is briefly introduced in Section II-A.

A. Interval Analysis

An interval $X_i = [\underline{x_i}, \overline{x_i}]$ is defined as the set of real x's such that $\underline{x_i} \leq x \leq \overline{x_i}$. The *width* $w(X_i)$ of the interval X_i is defined as $\overline{x_i} - \underline{x_i}$, while its *mid-point* is $(\overline{x_i} + \underline{x_i})/2$.

Here, we illustrate basic notions of interval analysis; interested readers may find a detailed explanation of the underlying theory in [21]–[24]. Interval analysis relies on interval arithmetics, the purpose of which is to determine guaranteed bounds for the minimum and maximum of a given function f over ranges for the unknowns with a minimal number of calculations. This determination is called an *interval evaluation* of the function and leads to a range $[\underline{F}, \overline{F}]$ that varies according to the ranges for the unknowns. If $\boldsymbol{X}$ denotes the ranges for the unknowns and $\boldsymbol{X_0}$ is a particular instance of the values of the unknowns within $\boldsymbol{X}$, then we have

$$\underline{F} \leq f(\boldsymbol{X_0}) \leq \overline{F}. \tag{4}$$

An interval evaluation may be calculated in different ways. The simplest is called the *natural evaluation*, which consists of using specific interval versions of all mathematical operators used in the function (interval version exists for all classical operators). For example, the addition of two intervals $\boldsymbol{a} = [\underline{a}, \bar{a}]$ and $\boldsymbol{b} = [\underline{b}, \bar{b}]$ is defined as $\boldsymbol{a}+\boldsymbol{b} = [\underline{a}+\underline{b}, \bar{a}+\bar{b}]$. Natural evaluation may be simply illustrated with $f = x^2 - 2x$ when x lies in the range [3, 5]. In that case, we can safely state that, for any instance of x in [3, 5], then x^2 lies in [9, 25], $2x$ in [6, 10] and, consequently, $-2x$ lies in $[-10, -6]$. Summing the interval for x^2 and $-2x$ leads to $[9, 25] + [-10, -6] = [-1, 19]$, which constitutes the interval evaluation of f over the range [3, 5]. Interval evaluation may also be obtained using other methods (e.g., Taylor expansion or centered form), but they will not be used in our algorithm. This example shows that simple operations are required by interval arithmetic, but also one of the drawbacks of the method. Indeed, clearly for any x in [3, 5], the value of f lies in [3, 15]: hence interval arithmetic overestimates the values of the minimum and maximum of the function. This occurs because we have multiple occurrence of the same variable in f which are considered to be independent during the calculation. However, this overestimation has the following properties:

- it does not always occur: for example, if f was defined as x^2+2x, then there will be no overestimation of the function for the range [3, 5];
- let us define an interval evaluation as $[\underline{a}, \bar{a}]$, f_m, f_M, the real minimum and maximum of the function over a given range and the size of the overestimation as $\text{Max}(f_m - \underline{a}, \bar{a} - f_M)$: in our example, the size of the overestimation is 4. However, this size decreases with the width of the input range. For example, assume that the input range for x is $[3, u]$: the size of the overestimation for f is 4 for $u = 5$, 2 if $u = 4$, 0.2 if $u = 3.1$ and becomes eventually 0 if $u = 3$

Furthermore, there are ways to decrease the size of the overestimation.

- An interval evaluation is sensitive to the analytical form of the function. For example, $f = x^2 - 2x$ may also be written as $f = (x-1)^2 - 1$, for which the interval evaluation is [3, 15], hence being exact. There is no known algorithm to determine the optimal analytical form of a function to be used by interval arithmetic, but the simple rule of rearranging the function to decrease the number of occurrences of the variable is usually efficient. For algebraic expressions, the use of Horner (or "nested") form is usually effective. For our example, f may be written as $x(x-2)$ and the natural evaluation of this form for $x = [3, 5]$ is [3, 15] and is hence optimal. The Horner form of f allows us to prove that there is no x in [3, 5], for which f will be strictly lower than 3.
- Derivatives and their interval evaluation may be used. In our example, the derivative of f is $2x - 2$, whose natural evaluation over [3, 5] is [4, 8]. This implies that the derivative is always positive over the range [3, 5], from which we deduce that the minimum and maximum of f will be obtained for x equal to 3 and 5, thereby leading to an exact interval evaluation.

An interesting property of interval arithmetic is that it can be implemented to take into account round-off errors. For example, if a calculation involves the number $1/3$, then there is clearly no computer floating-point number able to represent this number. There are two successive floating-point numbers f_1, f_2 such that $f_1 < 1/3$ and $f_2 > 1/3$ and, for any calculation involving this number, the computer will round it to the f_i that is the closest to $1/3$ (for instance, on a Pentium PC, the calculation of `3*(1/3.)-1` in C leads to -0.5510^{-16} instead of 0). In interval arithmetic, the number $1/3$ will be represented by the range $[f_1, f_2]$ so that any interval evaluation involving this number will always include the exact value of the calculation. Numerous packages of interval arithmetic are available, and our implementation of the singularity detection scheme is based on BIAS/PROFIL.[1]

Note that taking into account round-off errors allows one to safely compute the determinant at a given pose. If the lower bound of the interval evaluation of the determinant is greater than α or its upper bound is lower than $-\alpha$, then we may safely assert that the absolute value of the determinant is greater than α at this pose, which is the starting point of the algorithm as mentioned in the previous section. We must emphasize that round-off errors, which are often not considered in robotics, should be dealt with for critical applications and, if they are not so frequent, they may still occur even in simple cases. A classical example of this phenomena, due to Rump, may be observed when computing the floating-point value of

$$333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}$$

for $x = 77617, y = 33096$. Classical scientific software will return the value -10^{23}, interval evaluation computed in C is $[-0.56610^{23}, 0.55510^{23}]$, while the real value is ≈ -0.8273960599.

III. Algorithm Principle

Our singularity detection algorithm is based on a classical branch-and-prune algorithm, but uses efficient heuristics to drastically improve the computation time.

A *box* $\mathcal{B}$ is defined as a set of m ranges $\{I_1, \ldots, I_m\}$, one for each motion variety parameter t_i, i.e., to each box is associated a part of the variety. The mid-point or center of the box is the point whose coordinates are the mid-point of the ranges of the box. The box $\mathcal{B}_0$ is defined as the box for which all of the t_j have as range [0, 1], hence fully covering the variety. The width $w(\mathcal{B})$ of a box is defined as the largest width of its ranges.

We assume that we are able to calculate an interval evaluation $\mathcal{M}(\mathcal{B}) = [\underline{m}, \overline{m}]$ of $|\boldsymbol{J^{-1}}|$ for the ranges of $\mathcal{B}$ (the methods that may be used to calculate this interval evaluation are presented in Section III-A). A list of boxes $\mathcal{L}$ will be used by the algorithm: this list is indexed by k which will also be the iteration number of our algorithm, and $\mathcal{B}_j$ will denote the jth box of the list. When starting, the algorithm $\mathcal{L}$ has a single element $\mathcal{B}_0$, but, during the processing, boxes will be added to the list: r_k will be the total number of boxes in the list at iteration k.

Bisection of a box $\mathcal{B}_l = \{I_1^l, \ldots, I_m^l\}$ consists of choosing one of the motion parameters t_j and splitting its range $I_j^l = [a_j^l, b_j^l]$ into two ranges whose union will be I_j^l. It is usual to bisect I_j^l at the mid-point of the range, i.e., the two ranges resulting from the split are defined as $I_j^{l_0} = [a_j^l, (a_j^l + b_j^l)/2]$ and $I_j^{l_1} = [(a_j^l + b_j^l)/2, b_j^l]$. After having bisected the range I_j^l, the algorithm creates two new boxes $\mathcal{B}_l^0, \mathcal{B}_l^1$ whose ranges will be identical to the ranges of $\mathcal{B}_l$, except for the parameter t_j: for this parameter, the box $\mathcal{B}_l^0$ will have the range $I_j^{l_0}$ while the box $\mathcal{B}_l^1$ will have the range $I_j^{l_1}$ (hence, the union of $\mathcal{B}_l^0$, $\mathcal{B}_l^1$ is $\mathcal{B}_l$). They are different strategies for choosing the bisected variable [22], the simplest one being to choose the variable whose range has the largest width and its variant in which multiplicative weight is affected by the range widths, the selected variable being the one with the largest weighted width. Note that, according to the problem at hand, the choice strategy may drastically change the computation time [22], but, due to space constraints, we will not elaborate on this subject here.

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

The algorithm will return *SINGULARITY* if there is a box (i.e., a set of poses, part of the variety) for which $|\boldsymbol{J^{-1}}|$ is guaranteed to be lower than α for any pose in the set. If no such box is detected, it will return *POSSIBLE PROBLEM* if at a given pose we are unable to determine if $|\boldsymbol{J^{-1}}|$ is lower or greater than α, and otherwise the algorithm will return *NO SINGULARITY*.

We introduce a flag P that is initially set to 0 and which will be set to 1 if interval arithmetic is unable to determine if the determinant is greater than α for a given pose of the robot.

The basic idea of the algorithm is to split $\mathcal{B}_0$ into a set of boxes or, in other words, to separate the variety into small components until we may safely state the sign of the determinant for each pose in the variety component. Let us explain now the various steps of the algorithm.

- *Step 5*: interval evaluation of $|\boldsymbol{J^{-1}}|$ for the current box.
- *Steps 6–8*: for each pose of the motion variety associated with the current box, $|\boldsymbol{J^{-1}}|$ is guaranteed to be larger than α; no pose of type $\boldsymbol{X_2}$ may occur in this box. Hence the box is discarded.
- *Steps 10–11*: for each pose of the motion variety associated with the current box, we have $|\boldsymbol{J^{-1}}| < \alpha$ and the box is therefore a set of $\boldsymbol{X_2}$ poses. If $\overline{m}_k < 0$, we may even guarantee that any path between $\boldsymbol{X_1}, \boldsymbol{X_2}$ has to cross a singularity
- *Steps 15–18*: the width of the box is 0 (i.e., it is associated to one single pose of the motion variety), but round-off errors does not allow us to determine if $|\boldsymbol{J^{-1}}|$ is larger or lower than α. If the remaining of the process does not allow us to find a box with $|\boldsymbol{J^{-1}}| < \alpha$, then the result of the algorithm is uncertain.
- *Steps 20–28*: the interval evaluation of $|\boldsymbol{J^{-1}}|$ does not allow us to state if the determinant is larger than α for all poses of the motion variety associated with the current box. The current box is bisected in such way that the width of the box on top of the remaining boxes to be processed in $\mathcal{L}$ is always decreasing. This allows us to keep the storage size of $\mathcal{L}$ to a minimum. Formally, if the determinant of $|\boldsymbol{J^{-1}}|$ is larger than α when the width of the box is lower or equal to ϵ, then the maximal storage size of $\mathcal{L}$ should be $mE(\log(1/\epsilon)/\log(2)) + 1$, where $E(x)$ represents the integer closest to x. For example, if $\epsilon = 1e^{-6}$ and $m = 6$ we get that the storage size should be 120 boxes, i.e., $120 \times 6 \times 2 = 1440$ floating-point numbers. In practice, the storage size should be a little bit larger than this threshold, as when processing box k all boxes from 1 to k have already been processed but represent additional needed storage. A good practice is to delete the elements of the list with index from 1 to $k-1$ as soon as r_k is getting close to the maximum size of $\mathcal{L}$ so that the box at position k will become the first box of $\mathcal{L}$. One box resulting from the bisection is stored in place of the current box to save memory space, while the second one is stored according to its size either at the end of the list or immediately after

TABLE II
MODIFICATIONS OF ALGORITHM 1 TO DEAL WITH A MOTION VARIETY DEFINED BY $\mathcal{H}_j(\mathcal{T}) \leq 0$

4. **while** $(k \leq r_k)$ **do** 5. $\mathcal{M}(\mathcal{B}_k) = [\underline{m_k}, \overline{m_k}]$ 6. **if** $(\underline{m_k} > \alpha)$	4. **while** $(k \leq r_k)$ **do** 5. $\mathcal{M}(\mathcal{B}_k) = [\underline{m_k}, \overline{m_k}]$ 6. $\mathcal{H}_j(\mathcal{B}_k) = [\underline{s^j_k}, \overline{s^j_k}]$ $\forall j \in [1,p]$ 7. **if** $(\underline{m_k} > \alpha$ **or** $\exists \underline{s^j_k} > 0)$
10. **if** $(\overline{m_k} \leq \alpha)$	11. **if** $(\overline{m_k} \leq \alpha$ **and** $\forall j \in [1,p]\ \overline{s^j_k} \leq 0)$
14. **if** $(\underline{m_k} < \alpha$ **and** $\overline{m_k} > \alpha)$ **then**	15. **if** $(\underline{m_k} < \alpha$ **and** $\overline{m_k} > \alpha)$ **or** $(\exists\ \underline{s^j_k} < 0$ **and** $\overline{s^j_k} > 0)$

the current box (which requires moving box $\mathcal{B}_j$ to $\mathcal{B}_{j+1}$ for j in $[r_k, k+1]$; see steps 25–27).

- *Step 33*: if this step is reached, then all boxes in $\mathcal{L}$ have been processed. For each part of the variety associated with a box of the list, it has been shown that $|\boldsymbol{J^{-1}}| > \alpha$. As the whole motion variety has been covered, no singularity occurs within it
- *Step 35*: all boxes in the list have been processed, but during the processing a box with 0 width (i.e., a pose of the motion variety) has been found for which it was not possible to assert that $|\boldsymbol{J^{-1}}| < \alpha$ or $|\boldsymbol{J^{-1}}| > \alpha$ because of round-off errors. This has been detected at steps 16–18, and flag P has been set to 1. This means that the computer arithmetic does not allow a safe singularity detection; a multiprecision software must be used.

Such an algorithm is guaranteed to finish and to return either *SINGULARITY, NO SINGULARITY,* or *POSSIBLE PROBLEM*. The motion variety may also be defined by a set of p inequality constraints $\mathcal{H}_j(\mathcal{T}) \leq 0, j \in [1,p]$ on the motion parameters. Such a case may be managed as soon as we are able to calculate an interval evaluation of the inequalities and a box $\mathcal{B}_0$ that encloses the motion variety. Table II indicates which modification has to be done in Algorithm 1 to deal with such a case. The main differences are that we discard the boxes which do not belong to the variety (step 7) and bisect all boxes that are not guaranteed to be fully included in the variety.

The proposed method is a classical branch and prune algorithm, and we will discuss in the next sections improvements that may drastically influence the computation time, which are among the main contributions of this paper. As for any branch and prune method, the algorithm worst case complexity is exponential: it corresponds to the case where there is only a single pose $\mathbf{X}$ with $|\boldsymbol{J^{-1}}(\mathbf{X})| < \alpha$ in the motion variety. In practice however, such a situation does not occur frequently, and the results presented in Section V will show that the experimental complexity is rather low.

A. Determinant Evaluation

In Algorithm 1, only the interval evaluation of $|\boldsymbol{J^{-1}}|$ (step 5) is dependent upon the mechanical structure of the robot and upon the definition of the motion variety. Hence, provided that we are able to design an external module that is able to calculate such an interval evaluation, we have obtained a generic algorithm that is totally independent of the robot structure and from the motion variety; only the number of unknowns and their range will be needed.

There are different ways to obtain an interval evaluation of the determinant. In some cases, symbolic computation will allow to get a closed form of the determinant that may then be evaluated with interval arithmetic. In this paper, we will focus on the case where symbolic computation will not be able to provide a closed-form formula for the determinant (which may occur even for robots having a low number of DOF), although there is a closed form of the elements of $\boldsymbol{J^{-1}}$, allowing the calculation of their interval evaluation. After the interval, evaluations of its elements $\boldsymbol{J^{-1}}$ is obtained as an *interval matrix*. A classical linear algebra method such as row or column expansion or Gaussian elimination may then be used to compute the determinant of this interval matrix [25]. We have noted that, although the bounds were usually better with Gaussian elimination than with row or column expansion, the computational cost of Gaussian elimination is such that, overall, its use leads to a larger computation time. We thus preferably use the simplest form of row and column expansion through a recursive procedure that expands the determinant with respect to the row or the column having the lowest mean width. Note, however, that calculating the bounds for the determinant of $\boldsymbol{J^{-1}}$ based on its interval matrix representation will lead, in general, to a large overestimation of the determinant as the dependency between the elements of $\boldsymbol{J^{-1}}$ are completely ignored.

B. Improvement Heuristics

We present in the next sections some heuristics that may considerably improve the efficiency of the singularity detection scheme.

1) Matrix Conditioning: It is usually recognized that interval matrices appearing in a linear algebra problem should be *preconditioned*, i.e., that instead of using the matrix $\boldsymbol{J^{-1}}$ in the regularity check, we should use a matrix $\boldsymbol{KJ^{-1}}$ or $\boldsymbol{J^{-1}K}$, where $\boldsymbol{K}$ is a real nonsingular appropriately chosen matrix [25], [24]. The interest of preconditioning may be explained as follows. Consider that we have to find the bounds of the determinant of $\boldsymbol{J^{-1}}$ for all poses included in some box $\mathcal{B}_i$. We may calculate the real inverse Jacobian matrix $\boldsymbol{J^{-1}_{\mathrm{mid}}}$ for the pose corresponding to the mid-point of the box and evaluate safely its determinant with an interval evaluation $[\underline{a_{\mathrm{mid}}}, \overline{a_{\mathrm{mid}}}]$. If $\overline{a_{\mathrm{mid}}}$ is lower than α, then we have determined that the variety includes a pose close to a singularity (or a singularity if $\alpha = 0$), and the algorithm is completed. Let us assume that $\underline{a_{\mathrm{mid}}} > \alpha$ and calculate numerically an approximation $\boldsymbol{\tilde{J}_{\mathrm{mid}}}$ of the inverse $\boldsymbol{J_{\mathrm{mid}}}$ of $\boldsymbol{J^{-1}_{\mathrm{mid}}}$: this matrix will be be used as the preconditioning matrix $\boldsymbol{K}$ because it allows the product $\boldsymbol{KJ^{-1}}$ or $\boldsymbol{J^{-1}K}$ to be "closer" to the identity matrix and, therefore, more appropriate for an interval evaluation of its determinant. If S_K, S_{KJ} are the interval evaluations of the determinants of $\boldsymbol{\tilde{J}_{\mathrm{mid}}}$, $\boldsymbol{KJ^{-1}}$ (or $\boldsymbol{J^{-1}K}$), then S_{KJ}/S_K is an interval evaluation of the determinant of $\boldsymbol{J^{-1}}$, that may be used in Algorithm 1 instead of the direct evaluation of the determinant.

Usually, $\boldsymbol{K}$, $\boldsymbol{J^{-1}}$ and their products are evaluated numerically. However, a much more efficient approach is to assume first symbolic values for the elements of $\boldsymbol{K}$ for computing *symbolically* the product $\boldsymbol{KJ^{-1}}$ and then to use the analytical form of the elements of $\boldsymbol{KJ^{-1}}$ to evaluate the preconditioned matrix $\boldsymbol{J^{\text{cond}}} = \boldsymbol{KJ^{-1}}$ by plugging in the numerical values of the elements of $\tilde{\boldsymbol{J}}_{\text{mid}}$ as values for the element of $\boldsymbol{K}$. Let us illustrate why this approach is more efficient on the calculation of the element at the first row and first column J_{11}^{cond} of $\boldsymbol{J^{\text{cond}}}$ of a Gough platform (see Section V-B). This element is obtained as the product of the first row (k_{1j}) of $\boldsymbol{K}$ by the first column of $\boldsymbol{J^{-1}}$. The elements of this column may be written as $J_{j1}^{-1} = x_c + A_j$, where x_c is the x coordinate of the center of the platform and A_j is a term that depends upon the geometry of the robot, the orientation of the end-effector, and the leg number, but not of x_c. Hence, J_{11}^{cond} may be written in two different forms, called *form 1* and *form 2*, as

$$J_{11}^{\text{cond}} = \overbrace{\sum_{j=1}^{j=6} k_{1j}(x_c + A_j)}^{\text{form1}} = \overbrace{x_c\left(\sum_{j=1}^{j=6} k_{1j}\right) + \sum_{j=1}^{j=6} k_{1j}A_j}^{\text{form2}}.$$

In form 1, there are six occurrences of x_c, while there is only one in form 2, leading usually to a better interval evaluation for form 2. For example, if the first row of $\boldsymbol{K}$ is $(1, -1, 1, -1, 2, -1), x_c = [-1, 1]$ and $A_j = [-0.5, 0.5] \quad \forall j$, then the interval evaluation of J_{11}^{cond} with form 1 will be $[-10.5, 10.5]$ and only $[-4.5, 4.5]$ for form 2. Hence the symbolic preprocessing allows us to manage efficiently the dependency between the elements of $\boldsymbol{J^{-1}}$.

2) Interval Matrix Regularity Check: Here, we will assume $\alpha = 0$, i.e., we are trying to determine if a singular pose exists within the motion variety. For a given box $\mathcal{B}_k$, the interval evaluation of the elements of $\boldsymbol{J^{-1}}(\mathcal{B}_k)$ leads to an interval matrix $\boldsymbol{J_I^{-1}}$, i.e., a set of matrices that includes all inverse Jacobian matrices that may be obtained by instantiating $\boldsymbol{J^{-1}}$ with pose parameters belonging to $\mathcal{B}_k$, but also matrices that are not inverse Jacobian. The heuristic we will present in this section allows us to determine if the set $\boldsymbol{J_I^{-1}}$ includes a singular matrix. If for a given pose in $\mathcal{B}_k$ the determinant of $\boldsymbol{J^{-1}}$ is strictly positive and if the proposed heuristic allows us to determine that there is no singular matrix in $\boldsymbol{J_I^{-1}}$, then the determinant of all inverse Jacobian matrices obtained for a pose in $\mathcal{B}_k$ is positive: no pose of type $\boldsymbol{X_2}$ exists in $\mathcal{B}_k$, and this box may be discarded. A procedure based on this method will return `true` if $\mathcal{B}_k$ may be discarded and is used as a *filter* in step 6 of Algorithm 1.

The regularity test for an interval matrix that we will use has been proposed by Rohn [26]. We define H as the set of all n-dimensional vectors $\boldsymbol{h}$ whose components are either 1 or -1. For a given box, we denote by $[\underline{a_{ij}}, \overline{a_{ij}}]$ the interval evaluation of the component J_{ij}^{-1} of $\boldsymbol{J^{-1}}$ at the ith row and the jth column. Given two vectors $\boldsymbol{u}, \boldsymbol{v}$ of H, we then define the set of matrices $\boldsymbol{A^{uv}}$ whose elements $A_{ij}^{\boldsymbol{uv}}$ are

$$A_{ij}^{\boldsymbol{uv}} = \begin{cases} \overline{a_{ij}}, & \text{if} u_i.v_j = -1 \\ \underline{a_{ij}}, & \text{if} u_i.v_j = 1. \end{cases}$$

These matrices thus have elements with fixed numerical values (which are bounds of the interval evaluations of the elements of $\boldsymbol{J^{-1}}$), and there are 2^{2n-1} such matrices since $\boldsymbol{A^{uv}} = \boldsymbol{A^{-u,-v}}$. It may be shown that if the determinant of all of these matrices have the same sign, then all of the matrices $\boldsymbol{A'}$ whose elements have a value within the interval evaluation of J_{ij}^{-1} are regular [27]. Hence, for a 6×6 matrix $\boldsymbol{J^{-1}}$, if the determinant of 2048 scalar matrices has the same sign, then $\boldsymbol{J^{-1}}$ is regular for the current box. Note that we have proposed another regularity test that takes more into account the structure of the Jacobian matrix but which is more computer-intensive [28].

We may wonder if this theorem may be extended to the case where we have to show that $|\boldsymbol{J^{-1}}(\mathcal{B}_k)| > \alpha$ with α larger than 0. Let us define the set $\boldsymbol{A^e}$ of *extremal matrices* as the set of scalar matrices $\boldsymbol{J_e}$ derived from $\boldsymbol{J_I^{-1}}$ by taking as elements for $\boldsymbol{J_e}$ the lower or upper bound of the elements of $\boldsymbol{J_I^{-1}}$. There are 2^{n^2} such matrices, and the set $\boldsymbol{A^{uv}}$ is included in $\boldsymbol{A^e}$. It is easy to show that, if the determinant of all matrices in $\boldsymbol{A^e}$ is larger than α, then all matrices in $\boldsymbol{J_I^{-1}}$ also have a determinant larger than α. However, the number of matrices in $\boldsymbol{A^e}$ is so large (68 719 476 736 for a 6×6 matrix) that implementing a filter based on this result is not realistic. Determining if the Rohn test may be extended to this case is an open problem.

IV. Dealing With Uncertainties

Algorithm 1 may be adapted to take into account uncertainties occurring for two main reasons:

- errors in the robot geometrical modeling: the geometry of the robot is used for computing the inverse Jacobian matrix, but the geometry of a real robot will differ from its theoretical model as manufacturing errors will always occur;
- errors in the execution of the robot motion: there will always be deviations between the prescribed motion variety and the robot real motion due to control errors.

The influence of the manufacturing tolerances may be illustrated for a Gough platform whose dimensions are given in Section V-B. We have introduced an uncertainty of ± 0.1 in the coordinates of all the anchor points, and then we have computed the exact minimal and maximal values of $|\boldsymbol{J^{-1}}|$ at various poses. Compared with the value of $|\boldsymbol{J^{-1}}|$ calculated without uncertainties, relative variations of 25%–40% have been observed. Much larger variations may even be observed if we do not consider the exact minimal and maximal values of $|\boldsymbol{J^{-1}}|$ but its interval evaluation. For example, for the same amount of uncertainty, the exact minimal and maximal values of $|\boldsymbol{J^{-1}}|$ at a given pose were established as 108 571 and 176 505 while its interval evaluation was calculated as $[-451401, 692270]$.

Hence, if the uncertainty ranges are relatively large, Algorithm 1 may not be able to determine if the variety is singularity-free as, at most poses, we will be unable to determine if $|\boldsymbol{J^{-1}}| > \alpha$, and consequently the algorithm will return *POSSIBLE PROBLEM*.

In that case, it will be necessary to consider the elements affected by uncertainty as additional variables. A drawback is that symbolic computation tools may be no more able to calculate a closed form of the determinant (this is exactly what occurs with the robots presented in Section V-B). Consequently, we will have to rely on the method presented in Section III-A to calculate the interval evaluation of the determinant which will largely overestimate the bounds of $|\boldsymbol{J^{-1}}|$. Still, only the number of variables and the calculation of the interval evaluation of $|\boldsymbol{J^{-1}}|$ in

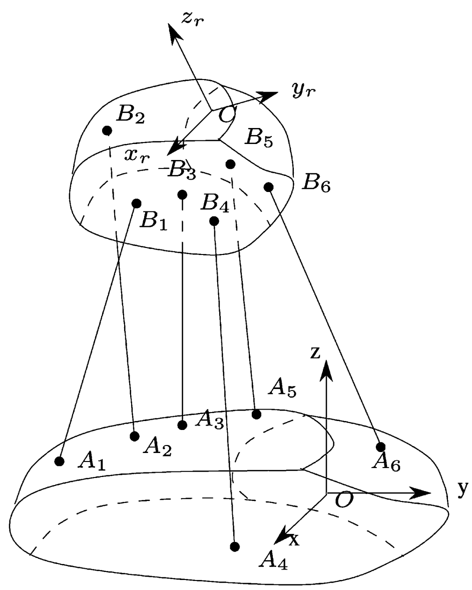


Fig. 1. Gough platform.

Algorithm 1 (step 5) have to be modified to deal with the uncertainties, and Section V will show that Algorithm 1 is able to tackle large uncertainties.

V. Examples and Computation Time

A. Implementation

Algorithm 1 and its variants that are allowed to deal with uncertainties have been implemented using our C++ interval analysis library ALIAS,[2] which provides already an implementation of Algorithm 1 for $\alpha = 0$. An interest of this library is that it is possible to use it directly within Maple. In our case, the end-user writes a Maple program that calculates J^{-1} and then calls a specific procedure that generates the necessary C++ code, compiles it, and then runs the created executable, which returns the result to Maple. Expert end-users may customize the C++ code within Maple and may even provide their own regularity heuristics.

This implementation allows one to perform the singularity check with minimal effort. Note that computing $\mathbf{J^{-1}}$ and writing the corresponding C++ code are necessary regardless of which method is used for the singularity check. Accordingly, comparison between methods should be based only on the execution time of the regularity check: this is presented in the following sections.

B. Robot

For the examples, we consider Gough-type 6-DOF parallel manipulators (Fig. 1): checking singularities for this type of robot is probably among the most challenging task as this robot has a full 6×6 inverse Jacobian matrix (robots with less than 6 DOF will have some 0 elements in their Jacobian matrix). The robot is constituted of a fixed base plate and a mobile plate connected by six articulated and extensible links. The pose of the platform may be 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 a universal joint, whose center is B_i. For a given robot, the components of $\boldsymbol{OA_i}$ in the reference frame are assumed to be known, possibly with some uncertainties. Similarly, the components of $\boldsymbol{C}B_{\boldsymbol{i}}^r$ in the mobile frame are assumed to be known. Let ρ_i be the joint variables (the distance between A_i and B_i), and $\boldsymbol{X}$ be a 6-D vector defining the pose of the end-effector. A possible parameterization for $\boldsymbol{X}$ is to use the reference frame coordinates x_c, y_c, z_c of C as three first components of $\boldsymbol{X}$, while the three last components are three parameters describing the orientation of the end-effector that allows one to calculate the rotation matrix $\boldsymbol{R}$ between the mobile and reference frame. In the examples, we use the Euler angles ψ, θ, ϕ, but any other representation may be used. Using this notation, we may compute the components of $\boldsymbol{CB_i}$ in the reference frame as $\boldsymbol{RC}B_{\boldsymbol{i}}^r$, while we have

$$\boldsymbol{A_iB_i} = \boldsymbol{A_iO} + \boldsymbol{OC} + \boldsymbol{RC}B_{\boldsymbol{i}}^r.$$

The length ρ_i of the ith leg is such that $\rho_i^2 = ||\boldsymbol{A_iB_i}||^2$ and can thus be calculated being given $\boldsymbol{X}$. It is well known that the ith row of the inverse Jacobian matrix of Gough platform is

$$J_i^{-1} = ((\frac{\boldsymbol{A_iB_i}}{\rho_i} \quad \boldsymbol{CB_i} \times \frac{\boldsymbol{A_iB_i}}{\rho_i})). \tag{5}$$

Hence, the inverse Jacobian matrix can be calculated being given $\boldsymbol{X}$. Note that if we define the row of a 6×6 matrix $\boldsymbol{M}$ as

$$\mathrm{M}_i = ((\boldsymbol{A_iB_i} \quad \boldsymbol{CB_i} \times \boldsymbol{A_iB_i})) \tag{6}$$

we may see that

$$|\boldsymbol{J^{-1}}| = \frac{|\boldsymbol{M}|}{\prod_{i=1}^{i=6} \rho_i}. \tag{7}$$

Therefore, the singular configuration may also be defined as the configuration for which $|\boldsymbol{M}| = 0$. It will be better to use this formula for singularity, as the overestimation of the matrix elements due to interval analysis will be lower compared to the elements of $|\boldsymbol{J^{-1}}|$.

We have considered two robot geometries for the tests. The coordinates of the leg attachment points A_i, and B_i of both robots are presented in Table III (in this section, all coordinates and length are expressed in centimeters).

C. Results

All tests are performed on a Dell D400 laptop (1.7 GHz), and all computation times are given in seconds. In all cases, we have assumed that it was not possible to determine a closed form of the determinant. We will present in various tables the computation time of Algorithm 1, denoted as Reg in the table, and combinations of this algorithm with heuristics: Rohn check

[2][Online]. Available: www.inria.fr/coprin/logiciels/ALIAS/ALIAS.html

TABLE III
COORDINATES OF THE A AND B POINTS FOR ROBOTS 1 AND 2

	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 IV
COMPUTATION TIME IN SECONDS FOR CHECKING THE 6-D WORKSPACE OF ROBOT 1 DEFINED BY x_c, y_c IN $[-5, 5]$, z_c IN [45, 50] ACCORDING TO THE ORIENTATION RANGES AND THE HEURISTICS USED IN THE ALGORITHM

Orientation range	Reg	Reg +Rohn	$J^{-1}K$	$J^{-1}K$ + Rohn	KJ^{-1}
[-1,1]	106.74	0.37	1.09	0.39	0.01
[-2,2]	338.9	0.4	4.3	0.42	0.01
[-5,5]	9076.2	2.6	34.79	2.8	0.01

(Section III-B2), denoted Rohn in the table, left matrix conditioning, denoted KJ^{-1} in the table, right conditioning, denoted $J^{-1}K$ (Section III-B1).

1) Singularity in 6-D Workspaces:

a) Exact Geometrical Robot's Model: For robot 1, we first consider a 6-D workspace defined by the following ranges: x_c, y_c in $[-5, 5]$ and z_c in [45, 50]. We first assume a perfect knowledge of the geometry of the robot and check this workspace for singularity for various orientation ranges using the basic algorithm and various heuristics: the computation times are presented in Table IV. This table shows that the KJ^{-1} conditioning is by far the most efficient. Note that the computation time is much higher if, instead of using a symbolically precomputed KJ^{-1}, $J^{-1}K$, we have calculated numerically $J_{\text{mid}}J^{-1}$ (or $J^{-1}J_{\text{mid}}$). For example, for the orientation range $[-5, 5]$, the use of $J_{\text{mid}}J^{-1}$ leads to a computation time of 70.22 s (instead of 0.01) and 79.13 s for $J^{-1}J_{\text{mid}}$ (instead of 34.79).

The efficiency of the use of the KJ^{-1} conditioned matrix is confirmed if we enlarge the 6-D workspace. For a 6-D workspace defined by the ranges x_c, y_c in $[-15, 15]$, z_c in [45, 50] and the orientation angles ψ, θ, ϕ in $[-15, 15]$ degree, no singularity is detected in this workspace in a computation time of 0.19 s (6028 s for Reg + Rohn).

If we enlarge the ranges for the orientation angles to $[-40, 40]$ a singularity is detected in the workspace in 0.25 s for Reg, 0.27 s for Reg + Rohn, 0.32 s for KJ^{-1} and $J^{-1}K$.

For robot 2, the workspace is defined by x_c, y_c in $[-200, 200]$, z_c in [2800, 3200]. Table V presents the computation time for the singularity check for various ranges for the orientation angles, assuming no uncertainty in the location of the anchor point. Note that, for the range $[-40, 40]$ degrees, the workspace includes a singularity.

TABLE V
COMPUTATION TIME (S) FOR THE SINGULARITY CHECK OF ROBOT 2, WORKSPACE DEFINED AS x_c, y_c IN $[-200, 200]$, z_c IN [2800, 3200], FOR VARIOUS RANGES FOR THE ORIENTATION ANGLES AND NO UNCERTAINTY IN THE LOCATION OF THE ANCHOR POINTS

Orientation range	Reg	Reg +Rohn	$J^{-1}K$	KJ^{-1}	KJ^{-1} +Rohn
[-20,20]	12.66	0.43	1.9	0.11	0.43
[-30,30]	94.31	2.9	11.88	0.94	1.9
[-40,40]	0.1	0.1	0.13	0.13	0.24

TABLE VI
COMPUTATION TIME (S) OF THE SINGULARITY DETECTION SCHEME WITH THE KJ^{-1} PRE-CONDITIONING, FOR ROBOT 1, VARIOUS 6-D WORKSPACE AND UNCERTAINTIES ϵ ON THE COORDINATES OF THE ANCHOR POINTS

x_c, y_c / z_c / ψ, θ, ϕ	$[-5, 5]$ / [45, 50] / $[-5°, 5°]$	$[-5, 5]$ / [45, 50] / $[-15°, 15°]$	$[-15, 15]$ / [45, 50] / $[-15°, 15°]$
(D) $\epsilon = \pm 0.05$	0.01	0.23	5.5 (7.32 with Rohn)
(V) $\epsilon = \pm 0.05$	0.01	0.63	14.07 (4.54 with Rohn)
(D) $\epsilon = \pm 0.1$	0.01	4.47	1540.74 (514.5 with Rohn)
(V) $\epsilon = \pm 0.1$	0.02	2.55	2614.55 (402.2 with Rohn)

b) Uncertainties in the Geometrical Model: We assume now that we have uncertainties on the location of the anchor points A_i, B_i. We may introduce directly these uncertainties in the inverse Jacobian (a method denoted by (D) in the table) or add these uncertainties as unknowns in the Jacobian elements (denoted by (V) in the table). Table VI presents the computation time of the singularity detection scheme using the KJ^{-1} preconditioning for various workspaces and uncertainty amplitude ϵ.

The use of the Rohn test is interesting when the computation time is relatively large, and having the uncertainties as additional variables is usually more interesting if the KJ^{-1} preconditioning is used in combination with the Rohn test.

Note that we have used a box model for the uncertainties, i.e., the real location of the A_i, B_i is assumed to lie within a 3-D box centered at the nominal location of these points. However, others models may be used as well: for example, we may assume that the anchor points are located within spheres centered at the nominal location (x_i^n, y_i^n, z_i^n) of the anchor points and known radii R_i by writing the Cartesian coordinates x_i, y_i, z_i of the anchor points as $x_i^n + r_i \sin(\alpha_i)\sin(\beta_i), y_i^n + r_i \sin(\alpha_i)\cos(\beta_i), z_i^n + r_i \cos(\alpha_i)$ and assigning the range $[0, R_i]$ to r_i, $[0, 2\pi]$ to β_i, and $[0, \pi]$ to α_i.

For robot 2, if we add uncertainties on the location of the anchor points directly in the elements of the inverse Jacobian, we find out that the workspace is singularity-free in almost a constant time of 0.12 s for an uncertainty up to a range of ±5. For an uncertainty of ±10, the computation time is 0.36 s and the workspace is still singularity-free. Starting around ±15, the algorithm finds poses for which it cannot determine the sign of the determinant. If we add the uncertainties as additional unknowns, it is then determined that the workspace is singularity-free for an uncertainty of ±20 in 5.47 s and in 388 s for an uncertainty of ±30.

These tests show that Algorithm 1 is rather efficient, even for large 6-D workspace, especially as it will allow us to avoid any real-time singularity check for trajectories performed in the 6-D workspace.

TABLE VII
COMPUTATION TIME (s) FOR CHECKING TRAJECTORIES T_1, T_2 ACCORDING TO UNCERTAINTIES AND HEURISTICS

Trajectory, uncertainties	Reg	Rohn	KJ^{-1}	$J^{-1}K$
$T_1, \epsilon = \epsilon_s = \epsilon_o = 0$	0.19	0.19	0.07	0.12
$T_1, \epsilon = 0.05, \epsilon_s = 0.01, \epsilon_o = 0.001$	0.51	0.29	0.01	0.22
$T_2, \epsilon = \epsilon_s = \epsilon_o = 0$	0.20	0.19	0.04	0.17
$T_2, \epsilon = 0.05, \epsilon_s = 0.01, \epsilon_o = 0.001$	15.52	2.22	0.27	2.3

2) Singularity on 1-D Trajectories: Two typical trajectories have been tested for robot 1. The first trajectory T_1 is a circular trajectory in the $x - y$ plane with a constant orientation. Formally, it is defined as

$$\begin{aligned} x_c &= 8\cos(2\pi T) \\ y_c &= 8\sin(2\pi T) \\ z_c &= 55 \\ \psi &= \theta = \phi = 0 \end{aligned}$$

where T is the time variable and lies in the range [0, 1]. The second trajectory T_2 has a varying orientation

$$\begin{aligned} x_c &= 8\cos(2\pi T) \\ y_c &= 8\sin(2\pi T) \\ z_c &= 55 \\ \psi &= 2\pi T \\ \theta &= \frac{5\pi}{180} \\ \phi &= -\psi. \end{aligned}$$

In all of tests, the interval evaluation of the determinant of $\boldsymbol{J}^{-1}$ is computed based on the elements of the matrix, although it was possible to obtain a closed form for the determinant if we assume that there is no uncertainty on the location of the anchor points and on the trajectory that is followed (for T_1, the determinant is constant, and for T_2 it is a polynomial in $\sin(2\pi T), \cos(2\pi T)$). Note that, if the location of the anchor points are given as symbolic values, then `Maple` is no longer able to calculate the closed form of the determinant.

The computation time of the singularity detection scheme for both trajectories, according to the amplitude ϵ of the uncertainty on the coordinates of the anchor points, the control errors ϵ_c on x_c, y_c, z_c and ϵ_o on ψ, θ, ϕ, and the heuristics are presented in Table VII. For trajectory T_1, the uncertainty ranges are directly added to the elements of the inverse Jacobian while for trajectory T_2 a similar approach leads the algorithm to return *POSSIBLE PROBLEM*. In that case, we must add the uncertainties on ψ, θ, ϕ as unknowns. Note that in all cases the trajectories are singularity-free.

These tests show that, even for a relatively complex trajectory, checking is real time if uncertainties are not taken into account and may be performed offline in a very reasonable time for relatively large and reasonable uncertainties.

VI. Conclusion

The algorithms proposed in this paper enable us to solve very important problems for the practical use of parallel robots. Symbolic preprocessing allows one to use generic algorithms that allow us to deal with any type of robot and with almost any type of motion variety, while the chosen numerical approach allows one to deal with uncertainty in the geometry of the robot and in the control. As may be seen from the results, the computation time is in general quite acceptable. It may be further reduced by using a distributed implementation (all interval analysis-based algorithms are appropriate for such an implementation). Prospective work will involve taking more into account the structure of the elements of the inverse Jacobian matrix in order to reduce the overestimation of the determinant.

References

[1] I. Bonev and D. Zlatanov, "The mystery of the singular SNU translational parallel robot," Jun. 2001 [Online]. Available: www.parallemic.org/Reviews/Review004.html

[2] R. D. Gregorio and V. Parenti-Castelli, "Mobility analysis of the 3-UPU parallel mechanism assembled for a pure translational motion," *ASME J. Mech. Design*, vol. 124, no. 2, pp. 259–264, Jun. 2002.

[3] R. Ben-Horin and M. Shoham, "Singularity analysis of a class of parallel robots based on Grassmann-Cayley algebra," in *Proc. Computat. Kinemat.*, Cassino, Italy, May 4–6, 2005, pp. 1–12.

[4] D. Downing, A. Samuel, and K. Hunt, "Identification of the special configurations of the octahedral manipulators using the pure condition," *Int. J. Robot. Res.*, vol. 21, no. 2, pp. 147–159, Feb. 2002.

[5] E. Staffetti, "Kinestatic analysis of robot manipulators using the Grassmann-Cayley algebra," *IEEE Trans. Robot. Autom.*, vol. 20, no. 2, pp. 200–210, Apr. 2004.

[6] B. M. St-Onge and C. Gosselin, "Singularity analysis and representation of spatial six-DOF parallel manipulators," in *Proc. ARK*, Portoroz-Bernadin, Slovenia, Jun. 1996, pp. 389–398.

[7] J.-P. Merlet, "Singular configurations of parallel manipulators and Grassmann geometry," *Int. J. Robot. Res.*, vol. 8, no. 5, pp. 45–56, Oct. 1989.

[8] S. Bhattacharya, H. Hatwal, and A. Ghosh, "Comparison of an exact and an approximate method of singularity avoidance in platform type parallel manipulators," *Mechanism Mach. Theory*, vol. 33, no. 7, pp. 965–974, Oct. 1998.

[9] B. Dasgupta and T. Mruthyunjaya, "Singularity-free path planning for the Stewart platform manipulator," *Mechanism Mach. Theory*, vol. 33, no. 6, pp. 711–725, Aug. 1998.

[10] A. Dash, "Instantaneous kinematics and singularity analysis of three-legged parallel manipulators," *Robotica*, vol. 22, no. 2, pp. 189–203, Mar. 2004.

[11] A. Dash, "Workspace generation and planning singularity-free path for parallel manipulators," *Mechanism Mach. Theory*, vol. 40, no. 7, pp. 778–805, Jul. 2005.

[12] C. Jui and Q. Sun, "Path tracking of parallel manipulators in the presence of force singularity," *ASME J. Dyn. Syst., Meas. Control*, vol. 127, no. 4, pp. 550–563, Dec. 2005.

[13] D. Nenchev and M. Uchiyama, "Singularity-consistent path planning and control of parallel robot motion through instantaneous-self-motion type," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, Apr. 24–26, 1996, pp. 1864–1870.

[14] S. Sen, B. Dasgupta, and A. Mallik, "Variational approach for singularity-path planning of parallel manipulators," *Mechanism Mach. Theory*, vol. 38, no. 11, pp. 1165–1183, Nov. 2003.

[15] J.-P. Merlet, "A generic trajectory verifier for the motion planning of parallel robots," *ASME J. Mech. Design*, vol. 123, no. 4, pp. 510–515, Dec. 2001.

[16] J.-P. Merlet, "Determination of 6D workspaces of Gough-type parallel manipulator and comparison between different geometries," *Int. J. Robot. Res.*, vol. 18, no. 9, pp. 902–916, Oct. 1999.

[17] J.-P. Merlet, "An improved design algorithm based on interval analysis for parallel manipulator with specified workspace," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 2001, pp. 1289–1294.

[18] M. Park and J. Kim, "Kinematic manipulability of closed chains," in *Proc. ARK*, Portoroz-Bernadin, Slovenia, Jun. 1996, pp. 99–108.

[19] P. Voglewede and I. Ebert-Uphoff, "Measuring "closeness" to singularities for parallel manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 28–30, 2004, pp. 4539–4544.

[20] A. Wolf and M. Shoham, "Investigation of parallel manipulators using linear complex approximation," *ASME J. Mech. Design*, vol. 125, no. 3, pp. 564–572, Sep. 2003.

[21] E. Hansen, *Global Optimization Using Interval Analysis*. New York: Marcel Dekker, 1992.

[22] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*. Berlin, Germany: Springer-Verlag, 2001.

[23] R. Moore, *Methods and Applications of Interval Analysis*. Philadelphia, PA: SIAM, 1979.

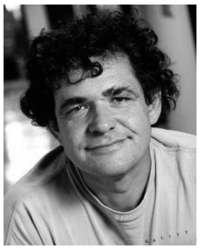
[24] A. Neumaier, *Interval Methods for Systems of Equations*. Cambridge, U.K.: Cambridge Univ. Press, 1990.

[25] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Boston, MA: Kluwer, 1998.

[26] G. Rex and J. Rohn, "Sufficient conditions for regularity and singularity of interval matrices," *SIAM J. Matrix Anal. Applic.*, vol. 20, no. 2, pp. 437–445, 1998.

[27] V. Kreinovich, "Optimal Finite Characterization of Linear Problems with Inexact Data," Univ. of Texas at El Paso, Tech. Rep. CS-00-37, 2000.

[28] J.-P. Merlet and P. Donelan, "On the regularity of the inverse Jacobian of parallel robot," in *Proc. ARK*, Ljubljana, Slovenia, Jun. 26–29, 2006, pp. 41–48.



J-P. Merlet (M'01) received the M.S. degree in mathematics from the University of Nantes, Nantes, France, in 1978, the Engineer title from the National Superior School of Mechanics, Nantes, in 1980, the Ph.D. degree from Paris VI University, Paris, France, in 1986, and the Research habilitation from Nice University, Nice, France, in 1993.

He has been an Engineer in the food industry (Lu) and in civil engineering. He has also been a Research Engineer with the CEA (French nuclear Agency) and a Research Associate with Kyoto University, MEL, Tsukuba, Japan, and with McGill University, Montreal, QC, Canada. He is now a scientific leader of the COPRIN project-team of INRIA (French National Research Institute in Control Theory and Computer Science). He has authored or coauthored over 200 conference and journal papers in the field of force control of robots, algebraic geometry, constraints solving, and parallel robots. He has published a book on the latter subject with two editions in French and two editions in English (Springer, 2001 and 205) He holds a patent on parallel robot which is currently licensed to a U.S. company (Gerber) and to a Japanese company (Hephaist Seiko). His current research interest are interval analysis, optimal design for mechanism, nanotechnology, and medical robotics.

Dr. Merlet is an Associate Editor of *Mechanism and Machine Theory* and Chairman of the French Section of IFToMM.