Int. J. Appl. Math. Comput. Sci., 2008, Vol. , No. , –
DOI:

amcs

INTERVAL ANALYSIS FOR CERTIFIED NUMERICAL SOLUTION OF PROBLEMS IN ROBOTICS

J-P. MERLET

INRIA,2004 Route des Lucioles, 06902 Sophia-Antipolis,France

Interval analysis is a relatively new mathematical tool that allows one to deal with problems that may have to be solved numerically with a computer. Examples of such problems are system solving and global optimization but numerous other problems may be addressed as well. This approach has the following general advantages:

1. it allows to find solutions of a problem only within some finite domain which make sense as soon as the unknowns in the problem are physical parameters
2. numerical computer round-off errors are taken into account so that the solutions are guaranteed
3. it allows one to take into account the uncertainties that are inherent to the parameters of a physical system

Properties 1 and 3 are of special interest in robotics problems, in which many of the variables are parameters that are measured (i.e. are known only up to some bounded errors) while the modeling of the robot is based on parameters that are submitted to uncertainties (e.g because of manufacturing tolerances). Taking into account these uncertainties is essential for many robotics applications such as medical or space robotics for which safety is a crucial issue. A further inherent property of interval analysis that is of interest for robotics problems is that this approach allows one to deal with the *uncertainties* that are unavoidable in robotics. Although the basic principles of interval analysis are easy to understand and to implement, this approach will be efficient only if the right heuristics are used and if the problem at hand is formulated appropriately. In this paper we will emphasize various robotics problems that have been solved with interval analysis, many of which are currently beyond the reach of other mathematical approaches.

Keywords: interval analysis, uncertainties, robotics

1. Introduction

We will consider in this paper robotized systems whose main purpose is to manipulate objects, although many other objectives may be assigned to such systems. A first important component of the robot is its *end-effector* which will grasp the object. The *pose* of the end-effector is defined as a set of parameters that allows one to determine what is the location/orientation of the end-effector in its surrounding world. For that purpose a reference frame $\mathcal{R}_f = (O, x, y, z)$ is defined, while a mobile frame $\mathcal{R}_m = (C, x_r, y_r, z_r)$ is attached to the end-effector. A possible set of parameters for defining a pose is first the three coordinates of the point C of the end-effector and three angles (such as the Euler's angles) that allows one to define a rotation matrix R between the vectors of the mobile frame and those of the reference frame. The end-effector is thus considered as a rigid body and it is well known that in the 3D space the minimal number of parameters necessary to define the location/orientation of this body is 6. The objective of a robot manipulator is to control all or part of the possible motion of the end-effector, called its *degree of freedom*. If a robot allows to control all possible motion of the end-effector it will be called a 6 degrees-of-freedom robot or 6 d.o.f. robot for short. For some tasks it is not necessary to control all motion: for example a crane that moves an object only along the x, y, z axis without offering the possibility of changing its orientation is a 3 d.o.f. robot.

In order to control the d.o.f. of the end-effector a robot manipulator has a mechanical structure, i.e. an arrangement of *joints* and *links*. A link is a rigid body that connect two (or more) joints. A joint allows for relative motion between two links that are connected to it. In

robotics the most frequently used joints allows only one possible type of motion between the links, for example a rotation around a given axis for *revolute joint* or a translation along a given axis for *prismatic joint*. Joints may be *passive* (they just follow the overall motion of the structure according to mechanical laws) or *actuated*: a motor is able to modify the relative position of the links that are connected to the joint. For actuated joints *sensors* are used to measure the relative motion of the links.

A typical robot manipulator is the *Scara* robot presented in figure 1. It has 4 d.o.f., allowing to move the end-effector along the x, y, z axis, but also to rotate it along the z axis.

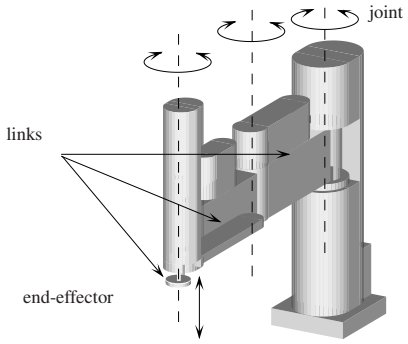

Fig. 1. The 4 d.o.f SCARA robot

Its mechanical architecture is called *serial*: starting from the ground we find a series of links and actuated joints. If we denote by **L** a link, by **R** a revolute joint and by **P** a prismatic joint, then the structure of the robot may be described as **LRLRLRLP**, the end-effector being connected to the extremity of the prismatic joint. All joints of this robot are actuated and it has no passive joint.

Hence a robot is a motion generator that allows one to modify the pose q of the end-effector (the objective) by adjusting the relative position θ of the links of the structure using the actuated joints (the control). As we will see most robotics problems involve the management of the relationship between q and θ (and possibly their time derivatives) under various constraints.

2. Robotics and certification

Certification is a crucial issue in robotics at different levels:

- *for a better understanding of the complex behavior of robotized systems*: simulations, even based on a theoretical model of the robot, should be able to present all aspects of the possible behavior of the robot. For example a robot may move among obstacles that have to be avoided and a simulation system should be able to detect all such collisions in spite of numerical round-off errors
- *for critical applications*: robots may have to perform safety-critical applications (e.g. medical robots performing surgical operations) and have thus to be certified, i.e. we have to ensure that even in the worst case the robot will behave correctly.

However, as every mechanically controlled system, uncertainties are an unavoidable element of a robotized system: we have manufacturing tolerances in the mechanical parts, sensor measurement errors, control errors, numerical round-off errors in the computer used for control and uncertainties in the surrounding world of the robot, to name a few. All these elements have to be taken into account when designing and building the robot and when controlling its motion.

Fortunately all these uncertainties have a common feature: they may be all *bounded*, i.e. we are able to determine intervals for each of them so that we are sure that the *real* value of a given parameter lie within the interval. Hence interval analysis is a tool that has to be considered when dealing a robotic problem. Interval analysis (Hansen, 2004),(Jaulin, Kieffer, Didrit and Walter, 2001),(Moore, 1979) is a numerical method that allows one to solve a broad range of problems (going from system solving to global optimization). In robotics it has been early used for solving the inverse kinematics problem (a problem that will be developed in the next section) for serial **6R** robot (Rao, Asaithambi and Agrawal, 1998) but is now used for addressing other robotic problems such as:

- the effect of clearance on the accuracy of robots (Wu and Rao, 2004),
- ensuring robot reliability (Carreras and Walker, 2001),
- mobile robot's localization and navigation (Ashokaraj et al., 2004), (Clerenti et al., 2003),(Kieffer, Jaulin, Walter and Meizel, 2000),(Seignez et al., 2005),
- planning the motion of robot (for example for avoiding obstacles) (Piazzi and Visioli, 2000),
- collision detection (Redon et al., 2004),
- calibration (i.e. find the real value of some geometrical parameters of the robot, the input being external measurements of the end-effector pose at various location) (Daney, Andreff, Chabert and Papegay, 2006)

to name a few. We will address in this paper some of these problems and will explain how interval analysis may provide a certified answer to them.

3. Interval analysis

In this special issue we will assume that the basic principles of interval arithmetic have been exposed. In practice for the implementation we are using the interval arithmetic package `BIAS/Profil`[1] that is widely distributed. Our algorithms will use interval boxes (i.e. a set of intervals) and we will assume that we are looking for a solution of a robotics problem only within a bounded domain, called the *search domain*, in the unknowns space. For the sake of simplicity we will assume that the search domain is also defined as a box, but this assumption may be dropped at will. In general an interval analysis algorithm may be described as the management of a list of boxes, each box in the list being submitted to 4 operators, namely filtering, evaluation, existence and bisection. We will now briefly describe the role of these operators when applied on a given box:

- *filtering*: this operator may show, in a certified way, that either the problem has no solution within the current box or that only a smaller box strictly included in the current box may contain solutions of the problem
- *evaluation*: this operator may show, in a certified way, that the problem has no solution within the current box or that all values of the unknowns within the current box are solutions of the problem
- *existence*: this operator may show, in a certified way, that there is a single solution of the problem in a box included in the current box, solution that may be calculated with an arbitrary accuracy
- *bisection*: this operator split the current box in two (or more) boxes by splitting one of the box interval into two (or more) intervals whose union is the initial interval

A box procedure manages the boxes list, which has a single element, the search domain, when starting the algorithm. It will discard from the list the boxes that have already been submitted to the operators or have been eliminated by the filtering or evaluation operators and add to the list the boxes resulting from the bisection operator. It will also store the solution as determined by the existence operator and the algorithm will complete whenever the list become empty. It may be seen that such algorithm is of the branch and bound type, whose worst case complexity is exponential because of the bisection process. However the practical complexity is quite often tractable, as will be seen later on.

We will now present some practical examples of the filtering, evaluation and existence operators, applied on a very simple example, finding the solutions of the equation $f(x) = x^2 - 4x + 1 = 0$ in the interval [-10,10].

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

3.1. Filtering. There are numerous methods that may be used for the filtering operator (Lebbah, Michel, Rueher, Merlet and Daney, 2004) but we will shortly describe a simple filtering approach, called the *2B* method. Equation $f(x) = 0$ may also be written as $4x = x^2 + 1$. Assuming that x has an interval value, this interval will include a solution only within the intersection of the interval evaluation of $4x$ and of $x^2 + 1$. If x is [-10,10], then this intersection is $[-40, 40] \cap [1, 101] = [1, 40]$. Assuming an interval evaluation of [1,40] for $4x$ we deduce that x should lie in the interval [1/4,10] while the inverse evaluation of $[1, 40] = x^2 + 1$ leads to $[-\sqrt{39}, \sqrt{39}]$ as possible value for x. Combining these two results we get that within the search domain only the interval $[1/4, \sqrt{39}]$ may include a solution of the equation. Hence with a few arithmetic operation we have been able to reduce the width of the search domain from 20 to less than 6. Note that we may repeat the procedure using the new interval for x by computing $[1, 4\sqrt{39}] \cap [17/16, 40] = [17/16, 4\sqrt{39}]$ but with a much smaller gain.

Such filtering method is called *local* because it deals with one equation and one variable at a time but there are also *global* methods (such as interval Newton) that may manage simultaneously several equations (Neumaier, 1990).

3.2. Evaluation. The most simple evaluation operator just consists in calculating the interval evaluation of the equation and determining if it includes 0. For example if we assume that x has the interval value [-10,-4], then the interval evaluation of $f(x)$ is [33,141] and we may safely discard this box as it cannot contain a solution of the equation. But more sophisticated evaluation operators exist, as will be presented later on.

3.3. Existence. We will now briefly introduce the Kantorovitch theorem that may be used to define an existence operator. Let a system of n equations in n unknowns:

$$f = \{f_i(x_1, \ldots, x_n) = 0, i \in [1, n]\}$$

each f_i being at least C^2. Let $\mathbf{x_0}$ be a point and a ball U, $U = \{\mathbf{x}/||\mathbf{x} - \mathbf{x_0}|| \leq 2B_0\}$, the norm being $||A|| = \mathrm{Max}_i \sum_j |a_{ij}|$. Assume that $\mathbf{x_0}$ is such that:

1. the Jacobian matrix of the system has an inverse Γ_0 at $\mathbf{x_0}$ such that $||\Gamma_0|| \leq A_0$
2. $||\Gamma_0 f(\mathbf{x_0})|| \leq B_0$
3. $\sum_{k=1}^{n} |\frac{\partial^2 f_i(\mathbf{x}}{\partial x_j \partial x_k}| \leq C$ for $i, j = 1, \ldots, n$ and $\mathbf{x} \in U$
4. the constants A_0, B_0, C satisfy $2nA_0B_0C \leq 1$

Then there is an unique solution of $f = 0$ in U and Newton method used with $\mathbf{x_0}$ as estimate of the solution will converge toward this solution (Tapia, 1971).

We will now illustrate how this theorem may be used to determine a ball centered at $x_0 = 4$, that will include a single solution of $x^2 - 4x + 1 = 0$. The Jacobian is $2x - 4$ whose inverse at x_0 is $A_0 = 1/4$, while $\Gamma_0 f(\mathbf{x_0}) = 1/4$, thus leading to $B_0 = 1/4$. The Hessian is constant and equal to $C = 2$. As $n = 1$ we get $2nA_0B_0C = 2 \times 1/4 \times 1/4 \times 2 = 1/4$. Consequently the Kantorovitch theorem is satisfied and we may conclude that there is a single solution of f in the interval [3.5, 4.5] (which indeed includes the solution $2 + \sqrt{3}$).

Note that a ball that includes a single solution of the system (denoted an *existence ball*) may be widened using the *inflation process* described by Neumaier (Neumaier, 2001). Let us assume that x_s is a solution of the system $f(x) = 0$ and consider a box $\mathcal{B}(x_s)$ centered at x_s. If J is the Jacobian of the system and for all points in $\mathcal{B}$ J is not singular, then the box includes only one solution of the system. As $\mathcal{B}$ is a box, calculating J for all points of the box leads to a set $\mathcal{S}$ of matrices. If we calculate now the interval evaluation of each element of J for $\mathcal{B}$ we get an *interval matrix* i.e. a set $\mathcal{I}$ of matrices, such that $\mathcal{S} \subset \mathcal{I}$ as the overestimation of the interval evaluation of the elements of J may lead to matrices that do not belong to $\mathcal{S}$. Consequently if we are able to show that the interval matrix does not include any singular one, then we can guarantee that x_s is the only solution of $f = 0$ in $\mathcal{B}$. Checking if an interval matrix does not include singular matrices may be performed using the following theorem:

let u be the diagonal element of a matrix H having the lowest absolute value, let v_i be the maximum of the absolute value of the sum of the elements at row i of H, discarding the diagonal element of the row, and let v be the maximum of the v_i's. If $u > v$, then the matrix is denoted *diagonally dominant* and H is regular.

This theorem may be extended to interval matrix by taking for u the lower bound of the absolute value of the interval diagonal elements of $\mathcal{I}$ and for v the upper bound of the interval valued v_i's. Note however that a preconditioning of the interval matrix $\mathcal{I}$ may be necessary for getting a stronger result: instead of applying the theorem on $\mathcal{I}$ we may use the interval matrix $J(x_s)^{-1}\mathcal{I}$, where $J(x_s)^{-1}$ is the inverse of the Jacobian at x_s. Assume now that Kantorovitch theorem has led to an existence box and that an approximate solution x_s has been calculated using the Newton scheme. If we define a "small" constant ϵ and a sequence of boxes centered at x_s as $[x_s - 2^m\epsilon, x_s + 2^m\epsilon], m \in [0, 1, 2, \ldots]$, then we may apply the regularity condition on each box of the sequence until it fails for $m = m_1$ and get a new existence box as $[x_s - 2^{m_1-1}\epsilon, x_s + 2^{m_1-1}\epsilon]$.

As soon as existence boxes have been determined we may use them for a filtering operator: if a box submitted to filtering has an intersection with an existence box, then we substitute it by its complement with respect to the existence box. Note however that it should be done only if this complement is a single box (or possibly a set of two boxes) as creating multiple new boxes may have a negative influence of the efficiency of the solving algorithm.

3.4. Bisection. When using the bisection process it is necessary to choose the unknown on which the bisection will be applied. This is a sensitive issue as this choice may drastically modify the running time of the algorithm. Classical choice methods are *largest first* (choosing the unknown having the interval value with the largest width) and *round-robin* (bisecting each variable in turn). The drawback of these methods is that they do not take into account the influence of the variable on the problem. Another method is based on the *smear function* introduced by Kearfott (Kearfott and Manuel, 1990). Let $J = ((J_{ij}))$ be the Jacobian matrix of the equations system and let define for each variable x_i the smear value $s_i = \mathrm{Max}(|J_{ij}[\underline{x_i}, \overline{x_i}]|, |\overline{J_{ij}[\underline{x_i}, \overline{x_i}]}|$. In the smear approach the bisected variable will be the one having the largest s_i. Our method of choice is to apply the smear function by default but to apply the largest first method after m iterations of the algorithm, m being a fixed integer that depends on the geometry of the problem.

3.5. General comments. We have presented in the above sections some fundamentals of interval analysis. Although the basic principles of interval analysis are pretty simple, it must be mentioned that in practice the implementation of an efficient interval analysis requires a high level of expertise. A very important issue is the way you define your problem: although mathematically equivalent the various forms are not so with interval analysis. This already appear in interval arithmetic as, for example, $x^2 + 2x + 1$ and $(x+1)^2$ are mathematically equivalent but will not always lead to the same interval evaluation; we will elaborate on that later on but a common mistake is to translate into interval analysis an already elaborated solution of the problem at hand instead of focusing on what is really the problem. Such a mistake may be illustrated by a request we have had from a colleague which provide us 3 very complex functions in three variables x, y, z, asking us to provide an approximation of the region in the variables space for which the 3 functions values were lying within some given interval. After a short discussion it has appeared that the functions were the closed-form solutions of a third order polynomial whose coefficients were simple x, y, z functions. Using the closed-form of the solutions it was almost impossible to determine the region as their interval evaluation has a very large width, even for almost point interval, while working with the polynomial was a trivial matter. Hence you must *think* in term of interval analysis and forget about other approaches.

Another issue is that the running time is heavily de-

pendent upon the right choice of heuristics that are used in the filtering, existence and bisection operator (an efficiency ratio of 1/100 000 can easily be obtained between a naive implementation and a sophisticated one). Unfortunately there is no known method allowing to determine what is the best combination of heuristics for a given problem.

This has motivated our development of the C++ `ALIAS` interval analysis library (Merlet, 2000) that includes a large number of heuristics and is combined with a `Maple` interface for an easier use.

We will now illustrate the use of interval analysis based algorithm on typical robotics problems.

4. Kinematics

4.1. Introduction. Kinematics is one of the first issue that has to be addressed when given a robot to control. The purpose is to establish the relationship between the pose parameters q of the end-effector and the actuated joint variables θ. We may distinguish two types of problems:

- *inverse kinematics*: being given a pose to be reached by the end-effector, what should be the corresponding joint variables ? this is the basic problem for control as the objective of a manipulator is to be able to reach a desired pose
- *direct kinematics*: being given the value of the joint variables (e.g. obtained through the sensors) what is (are) the possible corresponding pose (s) of the end-effector ? this is also a basic control problem as soon as the robot is controlled through a closed-loop scheme

To illustrate this problem we will consider a special robot structure called *parallel robot*. In a serial robot the end-effector is connected to the ground through a single kinematic chain, while in parallel robot several chains are used for the same purpose. A typical example of parallel robot is the Gough platform (Gough and Whitehall, 1962), shown in figure 2. In this robot the end-effector is the upper platform while the lower platform, (the *base*), is fixed. The end-effector is connected to the base through 6 identical chains, called the *legs* of the robot. Each chain is constituted by a passive spherical joint at A_i (which allow any rotation of the link around A_i), an actuated prismatic joint and a passive spherical joint at B_i. The attachment points of the leg on the base are in a known position in the reference frame, while the attachments points on the platform are in a fixed position that is known in the mobile frame. The joint variables of this robot are the 6 lengths ρ of the legs (that can be modified by controlling the motion of the prismatic joints). Hence solving the inverse kinematics of this robot amount to determine the 6 ρ for a given pose

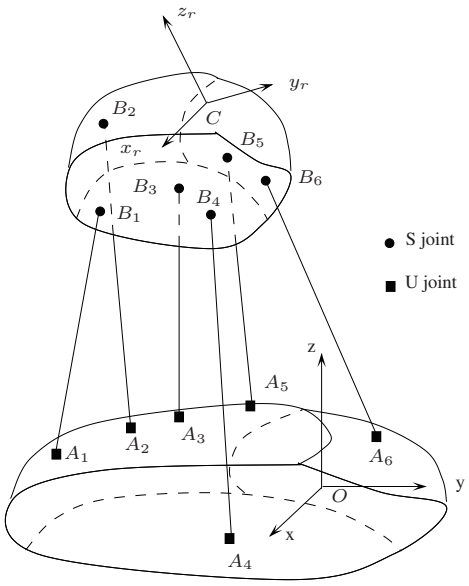

Fig. 2. Another possible mechanical structure for a robot: the Gough platform

of the mobile platform, while the direct kinematics is the problem of determining what are the possible poses of the mobile platform for given values of the 6 ρ.

Inverse and direct kinematics are dual problem for which the same set of equations is used, but whose unknowns will change according to the problem at hand. First we will establish the relationship between q and ρ and for that purpose we should note that for a given leg i the leg length ρ_i is the Euclidean norm of the vector A_iB_i. For now on we will drop the leg index as the formula that will be derived is identical for all legs. Using Chasles relation we get

$$AB = AO + OC + CB \tag{1}$$

As mentioned before the coordinates of B is known in the mobile frame and therefore the components of the vector CB is known in this frame. We will denote by CB_m this vector when its components are expressed in the mobile frame. If the rotation matrix $R(q)$ between the mobile frame is known, then the components of the vector CB in the reference may be obtained as $CB = R(q)CB_m$. Thus we have

$$\rho^2 = ||AB||^2 = ||(AO + OC + R(q)CB_m)||^2 \tag{2}$$

Equation (2) is the core equation that will be used for both inverse and direct kinematics. Note that in this equation we have components that are derived from the geometry of the robot (OA, CB_m), joints parameters (ρ) and elements that may be derived directly from q ($OC, R(q)$). For the inverse kinematics q is known and hence the right hand

side of (2) can be directly calculated, leading to the square of the joint variables. Consequently solving the inverse kinematics is straightforward. For the direct kinematics the 6 ρ_i^2 are known and we must determine the q that satisfies the 6 equations (2). This problem is quite difficult (it was qualified as "the Everest of modern kinematics" by F. Freudenstein, the father of this discipline). It may be shown that the problem may have up to 40 real and complex solutions (Ronga and Vust, 1992) and that there exists configuration with 40 real solutions (Dietmaier, June 29- July 4, 1998). As mentioned previously finding all solutions is important because the solution (i.e. the pose at which the end-effector is currently located) will be used for the robot control: missing the solution or, worth, choosing the incorrect one, may lead to catastrophic situations. If we assume that the core kinematic equations are algebraic (and the kinematic equations for the Gough platform may indeed be converted into such form) there are 3 possible methods to solve them:

- elimination method (Innocenti, 2001)
- continuation method (Wampler, 1996)
- Gröebner basis method (Rouillier, 1995)

The two first methods have merits but also a major drawback: they may miss solutions as they do not take into account round-off errors. The third method is, as interval analysis, *certified* in the sense that it cannot miss a solution and furthermore *exact* in the sense that it can provide the solutions with an arbitrary accuracy. The main limitation of the Gröebner basis method is that only rational coefficients may be used, thereby imposing in some cases to solve only an approximation of the real system.

The above methods have also a drawback: they compute all the possible solutions, although for the robotic problem we are only interested in the one that represents the *actual* pose of the platform. Currently there is no known method to sort out among the set of solutions which one corresponds to the actual pose. A second drawback is that it is almost impossible to use a priori knowledge on the solution within the solving scheme. For example physical joints have motion limits that will be incompatible with some theoretical solution of the kinematic equations, direct kinematics may have been solved a short time before the current calculation which allow to state that the current actual pose lie within some ball centered at the previous pose . . . All these information can only be used after the solving in order to eliminate incompatible solution and therefore they do not impact the solving time.

Furthermore direct kinematics may be used in a real-time context (i.e. the solution should be obtained as fast as possible). Typically a robot controller has a sampling time between 1 and 5 ms and the solving time should be less than this sampling step. But in that case, as the direct kinematics is solved at each sampling period, we may easily derive from the last obtained pose and the maximal velocities of the end-effector a relatively small ball $\mathcal{S}$ that must include the actual pose. This explains why the Newton scheme is used most of the time in this context. But this not a safe approach because we have no guarantee about the convergence of this method and furthermore it is well known that the Newton scheme may converge toward a solution that is no the closest from the initial guess. Another problem with the Newton scheme is that it is not able to manage the case where we have several solutions of the problem within $\mathcal{S}$, meaning that the obtained measurements do not allow to determine the actual pose. In such case the robot must be stopped immediately as we are no more able to control it safely. Hence a *certified method*, that is able to find all solutions within a given ball, and allowing one to incorporate additional knowledge, is needed.

4.2. Solving the direct kinematic with interval analysis.

4.2.1. Problem formulation. It can be seen that interval analysis may look like an appropriate tool for solving this problem. But as mentioned in the general comments we have to determine which form of the problem is the most appropriate. We have already exposed a possible form with a minimal number of parameters for the pose of the end-effector, but it has the drawback that multiple occurrences of the variables appears in the core equations. We will propose here another formulation that avoid this drawback, but increase the number of variables. For the sake of simplicity we will assume that the end-effector is planar, i.e. the B_i points all lie in the same plane. We will choose as variables of the problem the coordinates of three of the B_i points (called the *reference points*), say B_1, B_2, B_3, leading to a total of 9 unknowns. It is then easy to show that for the remaining B_i there exists a set of 3 constants $\alpha_i^k, k \in [1, 3]$ such that

$$OB_i = \sum_{k=1}^{k=3} \alpha_i^k OB_k \quad i \in [4, 6] \tag{3}$$

We may now write the 6 kinematics equations giving the square of the leg lengths as

$$\rho_i^2 = ||A_iO + OB_i||^2 \tag{4}$$

These six equations are basically distance equations that can be written as functions of the 9 variables. Among these equations the one obtained for leg 1 to 3 each involves only 3 variables. Furthermore each variable appears only once in the equations, thereby leading to an

optimal interval evaluation. Furthermore these equations are quite appropriate for the 2B filtering.

Three additional constraint equations are obtained by writing that the distance between each pair of points in the set $\{B_1, B_2, B_3\}$ is a fixed constant d_{ij}:

$$||B_iB_j||^2 = d_{ij}^2 \ \forall i, j \in [1,3], i \neq j \tag{5}$$

Note that each of these equations involves only 6 of the 9 variables and that, again, there is a single occurrence of the variables in the equations. Consequently we end up with a system of 9 quadratic equations in 9 variables and consequently the Jacobian matrix elements are linear in the variables, while the Hessian matrix is a constant matrix.

Another interest of this formulation is that all the variables may be bounded. Indeed in practice there are limits on the maximum length of the leg as a prismatic actuator can only extend up to a certain limit. Let us denote by ρ_{max}^i the maximal length of leg i and by d_i the distance between C and B_i. With this notation all the components of the vector A_iB_i are constrained to lie in the interval $[-\rho_{max}^i - d_i, \rho_{max}^i + d_i]$. If we consider now the components of the vector OB_i we may use the Chasles relation $OB_i = OA_i + A_iB_i$ to obtain bounds for the coordinates of the B_i as the components of OA_i are known. Furthermore it may be shown (Merlet, 2004) that the search domain obtained when considering individually each leg may be reduced if we consider a chain constituted by two legs of the platform (e.g. the chain A_1, B_1, B_2, A_2) as clearly the closed structure of this chain imposes more constraints on the motion of the B_i.

Note also that we may choose at will the reference points, this choice having an influence on the computation time of the solving. This may be seen when computing the bounds for the variables (i.e. the search space): selecting the legs whose absolute values for $\rho_{max}^i + d_i$ are minimal decreases the size of the search space. But the choice also influences the values of the α_j^i coefficients which play also a central role in the algorithm. In (Merlet, 2004) we have considered the computation time for all possible choices of the reference points and have shown that the ratio between the minimal and maximal computation time was about 28, an heuristic rule allowing to determine what is the best choice for a manipulator of given geometry.

4.2.2. Existence operator and the inflation process. The structure of the system we have to solve is quite special and allows one to specialize the theorems that are used in the general case. For example we have been able to show that for the Kantorovitch theorem this special structure allows one to substitute the n (number of equations, here $n = 9$) by the dimension of the ambient space (here 3), thereby leading to a wider existence box. We will now show that the inflation process may also be specialized so that instead of incrementally increasing the size of the existence box until the regularity condition does not held (which is computer intensive), we may directly compute the largest radius of the existence box.

We have seen that each components of the Jacobian matrix of the system are linear in terms of the unknowns. Let $\{x_i^0\}$ be the elements of X_0, J_0^{-1} the inverse of the Jacobian matrix computed at X_0 and let X_1 be defined as $\{x_i^0 + \kappa\}$, where κ is the interval $[-\epsilon, \epsilon]$. Each component J_{ij} of the Jacobian at X_1 can be calculated as $\alpha_{ij} + \beta_{ij}\kappa$, where α_{ij}, β_{ij} are constants which depend only upon X_0. If we multiply J by J_0^{-1} we get a matrix $U = J_0^{-1}J = I_n + A$, where I_n is the identity matrix of dimension n and A is a matrix such that $A_{ij} = \zeta_{ij}\kappa$ where the ζ_{ij} can be calculated as a function of the β coefficients and of the components of J_0^{-1}. For a given line i of the matrix U the diagonal element has a mignitude $1 - |\zeta_{ii}|\epsilon$ while the sum of the magnitude of the non diagonal element is $\epsilon \sum_{j=1}^{j=n} |\zeta_{ij}|,\ j \neq i$. The matrix U will be guaranteed to be regular if for all i:

$$\epsilon \sum_{j=1}^{j=n} |\zeta_{ij}| \ (i \in [1,n],\ j \neq i) \ \leq 1 - |\zeta_{ii}|\epsilon \tag{6}$$

which leads to

$$\epsilon \leq \frac{1}{|\zeta_{ii}| + \mathrm{Max}(\sum_{j=1}^{j=n} |\zeta_{kj}|), k \in [1,n], j \neq k} \tag{7}$$

Hence the minimal value ϵ_m of the right term of this inequality over the lines of U allows to define a box $[X_0 - \epsilon_m, X_0 + \epsilon_m]$ which contains an unique solution of the system. In general this box will be larger than the box computed with the Kantorovitch theorem.

4.2.3. Adding constraints. Physical constraints such as passive joint limits may allow to eliminate some of the theoretical solutions of the equations systems which violate this constraint. Such a constraint may easily be taken into account in the filtering operator. Consider for example the passive joint limits: typically a spherical joint has a *major direction* defined by a unit vector t and the angle between this direction and the direction of the leg that is connected to this joint cannot exceed a given limit λ. This constraint may be written as:

$$-\cos(\lambda) \leq \frac{A_iB_i.t}{\rho_i} \leq \cos(\lambda) \tag{8}$$

For a box in the interval analysis scheme we get ranges for the coordinates of B_i and it is easy to compute an interval evaluation $[\underline{a}, \overline{a}]$ of A_iB_i/rho_i. The current box may be eliminated if $\underline{a} > \cos(\lambda)$ or $\overline{a} < -\cos(\lambda)$. Furthermore the 2B method can be applied on both inequalities to reduce the size of the box.

4.2.4. Results and managing uncertainties. Extensive results are provided in (Merlet, 2004) and show that interval analysis is competitive with the fastest Gröebner basis method for providing all solutions (typically in a computation time ranging between 10 and 30 seconds). But as soon as additional constraints, such as joint limits, are introduced, interval analysis become the fastest available certified method. This is also true for the real-time context for which the interval analysis method, although presenting a computation time that is larger than the classical Newton scheme, remains compatible with the sampling rate of the robot controller while providing the right solution or detecting that multiple solutions lie within the search domain.

But there is an additional benefits in the use of interval analysis for this particular problem. All our calculation are based on a perfect knowledge of the physical parameters of the robot. In practice however we have bounded errors on the location of the A_i on the base, on the location of the B_i on the end-effector and on the leg lengths ρ as they are measured by a sensor that is inherently inaccurate. Still the core kinematics equations remains valid although its coefficients have now interval values. Consequently there is no more a finite number of solutions to the equations system but a solution region. Interval analysis may still be used in that case and will provide an inner and an outer approximation of this region, allowing to safely determine if the *real* robot presents kinematics performances that are compatible with the task at hand.

5. Singularities

We may now address an issue regarding parallel robots that is very important in practice. We consider the relationship between the end-effector velocities (translational and angular) and the actuated joint velocities $\dot{\theta}$. First we must mention that there is no pose parameters whose time-derivative correspond to the velocity vector of the end-effector. However for simplicity we will denote by $\dot{q}$ the 6 dimensional vector (v, Ω) that represents the translational and angular velocities of the end-effector. A well-known robotics property is that $\dot{\theta}$ and $\dot{q}$ are linearly related:

$$\dot{q} = J(q, \theta)\dot{\theta} \tag{9}$$

where the matrix J is dependent upon the pose of the end-effector and on the values of the joint parameters (active and passive). In the robotics literature this matrix is called the *Jacobian* of the robot although it is not a Jacobian in the mathematical sense. For a serial robot the matrix J can be simply derived from the structure of the robot, while for parallel robots it is usually easier to derive the inverse Jacobian matrix, that for simplicity we will denote by J^{-1}, so that

$$\dot{\theta} = J^{-1}(q, \theta)\dot{q} \tag{10}$$

An interesting property occurs when J^{-1} is singular: the end-effector velocity may not be 0 although the active joints are locked (i.e. $\dot{\theta} = 0$). Hence the robot may exhibit infinitesimal motion with locked actuators and hence the robot is no more controllable. The locations q, θ at which J^{-1} is singular are called the *singularities* of the robot.

But there is another property of singularities that is very important. For reaching a mechanical equilibrium the external forces and torques (summed up in the *wrench* $\mathcal{F}$) to which is submitted the end-effector must be compensated by the internal forces in the legs, that will be denoted τ. For a Gough platform the internal forces are directed along the leg and applied at point B_i on the end-effector. As there is a complete duality between wrench and velocities because of the virtual work principle, $\mathcal{F}$ and τ are linearly related:

$$\mathcal{F} = J^{-T}(q, \theta)\tau \tag{11}$$

Being given $\mathcal{F}$ the components of τ may be expressed as a ratio

$$\tau_i = \frac{|A_i|}{|J^{-T}|} \tag{12}$$

where A_i is the minor associated to τ_i. As $|J^{-T}|$ appears in the denominator, if the robot come close to a singularity the joint forces may go to infinity, leading to a breakdown of the robot. It is therefore important to check that the robot may not encounter a singularity within its work area (called a *workspace* in robotics) and this check will be addressed in the next section.

5.1. Checking workspace for singularity.

5.1.1. Principle. In the general case the inverse Jacobian matrix of a 6 d.o.f. robot is a 6×6 matrix and for a Gough platform the $i - th$ line J_i^{-1} of this matrix is written as:

$$J_i^{-1} = ((\frac{A_i B_i}{\rho_i} \quad \frac{CB_i \times A_i B_i}{\rho_i})) \tag{13}$$

Note that such a line is the *normalized Plücker vector* of the line associated to leg i. Although the matrix has an analytical form calculating the expression of its determinant leads to a huge expression that is not easy to manipulate. A geometrical analysis has shown that the inverse Jacobian will be singular only for specific respective position of the lines associated to the legs (Merlet, 1989) but this geometrical approach does not allow to determine if a given workspace is singularity free.

Assume now that the pose parameters have interval values, these intervals being possibly reduced to a point interval. The inverse Jacobian matrix is now an interval matrix and an interval evaluation of its determinant may

be calculated using interval extension of classical determinant calculation methods such as row or column expansion and Gaussian elimination.

The problem we want to address is determining if a given workspace (assumed here for simplicity to be defined as a box in the pose parameters space) is singularity-free. Note that the location of the singularity, if any, as it will be necessary to change the design of the robot. Consequently we are not interested in the singularity location.

We will first select an arbitrary pose q_1 within the workspace and compute the determinant of the inverse Jacobian at this pose. More exactly we are interested in the sign of the determinant at this pose and interval arithmetic is used to safely determine this sign. Note that if the interval evaluation of the determinant at a given pose has not a constant sign either the workspace will include singularities or we will not be able to state that the workspace is singularity-free without using a more accurate arithmetic. Let us assume that at q_1 the determinant is positive. As the determinant is a continuous function of the pose parameters if we are able to determine a pose q_2 at which the determinant is negative, then we can guarantee that any path path joining q_1 and q_2 has to cross a pose at which the determinant is 0, i.e. a singular pose. We may now design an interval analysis algorithm whose purpose is to determine q_2 poses or to show that q_2 poses do not exist within the workspace.

5.1.2. Operators. The evaluation operator is simple to design as interval arithmetic allows one to calculate the interval evaluation of the determinant of the inverse Jacobian for a given box, but, as usual, it will be preferable to use a pre-conditioned matrix (Kreinovich, Lakeyev, Rohn and Kahl, 1998). The special structure of the inverse Jacobian matrix also indicate that a symbolic step before preconditioning may lead to a better interval evaluation of the determinant. Indeed if x denotes the first component of the pose parameter, then the elements of the first column of J^{-1} may be written as $x + u_i$ where u_i has a value that depends only upon the orientation angles of the end-effector and upon geometrical features of the robot. If we use a pure numerical pre-conditioning by multiplying the interval matrix J^{-1} by a constant matrix $K = ((k_{ij}))$ to produce the pre-conditioned matrix J^c, then the element J^c_{11} of J^c will be calculated as $J^c_{11} = \sum k_{1j}x + \sum k_{1j}u_j$, which has 6 occurrences of the variable x. If we assume now that K is a symbolic matrix that will be numerical only later on, we may use symbolic simplification procedures to obtain $J^c_{11} = x\sum k_{1j} + \sum k_{1j}u_j$ which has only a single occurrence of x and consequently may have a significantly lower width than the former version.

Assuming now than an interval evaluation of the determinant has been obtained for the current box, if its lower bound is positive, then we may discard the box (as it cannot contain any q_2 pose) and if its upper bound is negative, then we will have shown that the workspace is not singularity-free as all poses in the box are q_2 pose. Finally if the algorithm has processed all the boxes in its list, then the workspace is singularity-free.

The filtering operator may use a regularity test proposed by Rex and Rohn (Rex and Rohn, 1998). We define H as the set of all n-dimensional vectors 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^{-1}_{ij} of J^{-1} at the *i-th* row and *j-th* column. Given two vectors u, v of H, we then define the set of matrices A^{uv} whose elements A^{uv}_{ij} are

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

These matrices have thus elements with fixed numerical values (which are upper or lower bounds of the interval evaluations of the elements of J^{-1}). There are 2^{2n-1} such matrices since $A^{uv} = A^{-u,-v}$. It may be shown that if the determinant of all these matrices have the same sign, then all the matrices A' whose elements have a value within the interval evaluation of J^{-1}_{ij} are regular (Kreinovich, 2000). Hence for a 6×6 matrix J^{-1}, if the determinant of the 2048 scalar matrices A^{uv} have the same sign, then J^{-1} is regular for the current box. Note that we have proposed another regularity test that takes even more into account the particular structure of the Jacobian matrix but which is more computer intensive (Merlet and Donelan, 2006).

As for the bisection process it is beneficial to carefully order the created boxes in the list. Indeed although the order is no importance when the workspace is singularity-free as all boxes will be processed, the ordering has a high influence when there is a singularity in the workspace: the sooner we process the boxes that include singularities, the sooner will the algorithm stop. To order the new boxes we calculate for each of them the interval evaluation of the determinant. If the lower bound of this evaluation is positive the box is not stored, while if the upper bound is negative we have found a box which has only q_2 poses. If the evaluation $[\underline{a}, \overline{a}]$ includes 0, then we store on top of the list the box that has the lowest $\overline{a}$ (if the determinant at q_1 has been negative we will store on top of the list the box having the lowest $|\underline{a}|$).

The worst situation for the algorithm is when the workspace includes a singular pose that is located exactly on the border of the workspace. To manage this problem we exchange the box on top of the list with the last box in the list after a fixed number of bisection of the algorithm. If some singular pose are located inside the workspace they will be more easily located than the pose on the border. It may however occurs that for all poses in the workspace the determinant is positive except for a

single pose at which the determinant is exactly 0. This problem may be managed by flagging boxes whose width is lower than a given threshold, discarding them (although they are stored) and then performing a local analysis of the flagged box when the algorithm completes.

5.1.3. Dealing with uncertainties. Clearly, properly dealing with singularity is safety-critical as parallel robot may be used as medical robots or for entertainment theater that accommodate the public. Hence modeling errors should also be taken into account. In this particular case the sources of uncertainty are possible manufacturing tolerances on the location of the A_i, B_i points.

There are two possibilities for dealing with these sources:

- leaving their interval values in J^{-1}. A consequence is that the determinant will always have an interval value. This may lead to a failure of the algorithm as at a given pose the interval evaluation of the determinant may not have a constant sign. However such failure can be detected and we may switch to the later option
- adding the coordinates of the A_i, B_i (or some of them) as new variables in the algorithm and therefore submitting them to the bisection process. This will significantly increase the computation time

However with this adaptation we get an *application certified* algorithm: if it returns that the workspace is singularity-free, then the *real* robot will also be singularity-free.

5.1.4. Results. We have considered a 6 d.o.f robot without uncertainty and tested various algorithm variants: using only the interval evaluation of the determinant (1), interval evaluation of the determinant with Rohn filtering (2), using symbolic post-conditioning of J^{-1} (3), applying symbolic post-conditioning of J^{-1} and Rohn filtering (4) and finally using symbolic pre-conditioning of J^{-1}. Typical computation time for these variants are presented in table 1. This table shows that symbolic pre-conditioning is by far the most efficient method. If we have a $[-\epsilon, \epsilon]$ interval uncertainty on all the coordinates of the A_i, B_i points, we get the computation time presented in table 2 for various workspaces (x, y, z are the coordinates of C, while ψ, γ, ϕ are the three orientation angles) and for various values for ϵ (that are compatible with classical manufacturing tolerances). In these test we have used symbolic pre-conditioning of J^{-1} and a (D) indicates that we have left the uncertainties in J^{-1} while a (V) indicates that they have been added as new variables. For the last workspace the time in parenthesis is obtained when using also the Rohn filtering. It may be seen that even with relatively large uncertainties it is not necessary to add new variables while the Rohn filtering shall be used as soon as they become large. It must be noted that in each case the tested workspace was singularity-free; if this is not the case the algorithm is much faster as the heuristic used to order the box in the list allows to determine quickly a box with only q_2 pose, avoiding the processing of the remaining boxes.

Algorithm	1	2	3	4	5
Time	9076.2	2.6	34.79	2.8	0.01

Table 1. Computation time in seconds for a regularity check of a robot without uncertainty

ϵ	$x, y \in [-5, 5]$ $z \in [45, 50]$ $\psi, \gamma, \phi \in [-5°, 5°]$	$x, y \in [-5, 5]$ $z \in [45, 50]$ $\psi, \gamma, \phi \in [-15°, 15°]$	$x, y \in [-15, 15]$ $z \in [45, 50]$ $\psi, \gamma, \phi \in [-15°, 15°]$
(D) ± 0.05	0.01	0.23	5.5 (7.32)
(V) ± 0.05	0.01	0.63	14.07 (4.54)
(D) ± 0.1	0.01	4.47	1540.74 (514.5)
(V) ± 0.1	0.02	2.55	2614.55 (402.2)

Table 2. Computation time for the regularity check for various workspaces and uncertainty $[-\epsilon, \epsilon]$ for the location of the A_i, B_i points

We have also investigated a variant of the proposed algorithm in which we want to detect if for a pose in the workspace the absolute value of $|J^{-1}|$ is lower than a fixed threshold. We are currently investigating a practical approach whose purpose is to determine the regions of the workspace in which the forces in the leg are lower, in absolute value, than a fixed threshold: this correspond exactly to an engineering problem in which each mechanical elements have a known breaking force, the robot having to avoid poses at which the force in a leg is larger than the minimal breaking force of the elements of the leg. We have exhibited an algorithm relying on algebraic geometry that is able to calculate the border of 2D cross-sections of the safe regions for a given wrench applied on the end-effector (Hubert and Merlet, 2008) but an extension of this algorithm to be able to deal with a set of wrenches and with uncertainties in the robot modeling will require the use of interval analysis.

6. Appropriate design

Up to now we have addressed problem that may be coined *analysis* problem: being given a robot (possibly with uncertainties) we have performed an analysis of its performances and have verified if they were in accordance with the requirements. But if the performance analysis shows that the robot does not comply with the requirements we have then to determine a new design for the robot. This

design area is coined a *synthesis* problem in which, starting from a general topology of the mechanical structure of the robot, we have to determine the geometrical parameters of the structure so that

- we may effectively build the robot
- the robot will comply with the requirements in spite of unavoidable uncertainties in its physical realization

6.1. Principle. A robot geometry is defined by a set of m parameters (that may be lengths, unit vector of rotation axis, inertia, ...) that are summed up in the vector $\mathcal{P}$. We define the m-dimensional *parameter space* as a space in which each dimension is associated to one element of $\mathcal{P}$. Hence a point in this space corresponds to a physical instance for the robot. For example for a parallel robot the vector $\mathcal{P}$ includes the coordinates of the attachment points A_i, B_i and possibly other parameters such as the minimal and maximal length of the legs. In practice note that to solve a synthesis problem we will not have to explore the whole parameters space: as the parameters have a physical meaning we may safely assume that there are bounded (e.g. the length of a robot link cannot be lower than 0 and has certainly an upper limit ...). Hence we may define a search domain in the parameters and only solutions within this search domain should be found for the synthesis problem.

A typical requirement from an end-user usually involve a minimal workspace $\mathcal{W}$ (the robot should be able to reach any pose within $\mathcal{W}$) and constraints that may be defined as

$$\forall q \in \mathcal{W} \;\; f(q) \leq 0, \;\; g(q) = 0 \tag{14}$$

where f, g are some explicit functions of q. For example if the leg lengths of a Gough platform over $\mathcal{W}$ should be lower than a given threshold ρ_{max}, then f will be the function $\rho^2(q) - \rho^2_{max}$ and there is no g constraint. On the other hand if the requirement is that the absolute value of the force τ in the legs over $\mathcal{W}$ should be lower than a given threshold τ_{max} for a given wrench $\mathcal{F}$, we cannot use the analytical form of τ as a function of $\mathcal{F}, q$ (which is very complex) and we will use instead

$$|\tau| - \tau_{max} \leq 0, \quad \mathcal{F} - J^{-T}(q)\tau = 0$$

Usually design algorithms in mechanical engineering relies on an optimization procedure that numerically determine a single value of $\mathcal{P}$ that minimize some real valued *cost function* that mixes all requirements (possibly with weights on each requirements) and are therefore called an *optimal design* approach. We have some reservations on these approach (e.g. that building the cost function is not an easy task whenever we have several requirements involving for example different units, without mentioning other drawbacks (Das and Dennis, 1997)). Our approach is instead based on the certified satisfaction of all requirements (14) and will provide not a single solution but a continuous set of solutions that will allow to manage uncertainties in the physical realization, as will be seen later on. Hence we have coined our methodology *appropriate design*.

Assuming that we have a single requirement, the constraints (14) define a set $\mathcal{R}$ of regions in the parameters space whose points all satisfy the constraints. An exact calculation of $\mathcal{R}$ is almost impossible except in very simple case. Furthermore computing exactly $\mathcal{R}$ may be considered as overkill: indeed points on the border of these regions are only *theoretical* solutions as designing a robot with such parameters may lead to a real robot whose representative point in the parameters space lie outside $\mathcal{R}$ and therefore violate the constraints. Consequently we aim at proposing an approach that is able to compute an *approximation* of $\mathcal{R}$ while ensuring that for all proposed theoretical solutions there will be a physical instance that will still satisfy the constraints.

6.2. Method for a single requirement. Starting from the search domain we may use an interval analysis algorithm $\mathcal{S}_1$ whose variables are the one of $\mathcal{P}$ and whose boxes will be denoted $\mathcal{B}_\mathcal{P}$. The evaluation procedure is somewhat special as it has also a structure of an interval analysis algorithm $\mathcal{S}_2$, whose variables are the one of $\mathcal{W}$ and whose boxes will be denoted $\mathcal{B}_\mathcal{W}$. This evaluation is in charge of ensuring that, being given the robot parameters interval values defined by the current box $\mathcal{B}_\mathcal{P}$, for all poses in $\mathcal{W}$ either (14) is satisfied or that for some box $\mathcal{B}_\mathcal{W}$ the constraint (14) will always be violated, thereby disqualifying $\mathcal{B}_\mathcal{P}$ as a design solution. If $\mathcal{S}_2$ completes, then $\mathcal{B}_\mathcal{P}$ is retained as a design solution.

Clearly if the width of the intervals in $\mathcal{B}_\mathcal{P}$ are large $\mathcal{S}_2$ will not complete. Hence we allow only a limited number of bisection in $\mathcal{S}_2$, that is inverse proportional to the width of $\mathcal{B}_\mathcal{P}$. If this number is reached $\mathcal{S}_2$ returns a signal to $\mathcal{S}_1$ that indicates that a bisection on the current $\mathcal{B}_\mathcal{P}$ must be performed.

We also impose a lower limit on the width on each element of $\mathcal{B}_\mathcal{P}$ for allowing a bisection on this element. For the current box a bisection will be allowed only on the variable of $\mathcal{P}$ who have an interval width larger than their threshold. If none of the interval satisfy this constraint, then the current $\mathcal{B}_\mathcal{P}$ is discarded. The threshold for each element of $\mathcal{P}$ is twice the error bounds that is assigned to the physical instance of the parameters. The motivation of this rule is that although $\mathcal{B}_\mathcal{P}$ may be a (or include) theoretical design solutions a physical instance of this solution, even designed with the center of the box as nominal values for the parameters, may lie outside of the box and therefore may violate the constraint (14).

The output of our algorithm is therefore a list of

boxes $\mathcal{B}_\mathcal{P}$ that defines closed regions. A post-processing determine for each box in this list if the box obtained by growing the box by the error bound on each parameter is still included in the region. If this is the case the box is definitely retained as a design solution, otherwise we decrease the box by the the error bound on each parameter and store the obtained box as a design solution. Consequently the final output, if any, is an approximation of $\mathcal{R}$ and provide only certified design solutions whose physical instance are guaranteed to satisfy (14).

6.3. Dealing with multiple requirements. Two methods may be used if the problem has several performance requirements. We may first determine the region $\mathcal{R}$ for each of the requirement and then compute the intersection of the results (which amount to computing the intersection of boxes) for obtaining a region for which all requirements are satisfied. This approach has the advantage that the size of $\mathcal{R}$ for each requirement indicates how difficult is the satisfaction of the requirements: if the final result is empty we may thus provide information on which requirement has to be relaxed. But this method has the drawback that it is computer intensive as all requirements are treated independently.

An alternate approach is to feed as search domain for dealing with a given requirement the result of a previous run for another requirement. Dealing in sequence with requirement 1, 2, ... allows in general to decrease the size of the search domain at each step, thereby speeding up the computation. Furthermore there is no need to compute any intersection as the final result is guaranteed by construction to satisfy all constraints. A drawback appears if the final result is empty as there is no way to determine which requirements should be relaxed to get a result.

6.4. Limits and results. The proposed approach has the advantage of providing multiple solutions, thereby allowing a final choice that may be based on various criterion, including economical one. But the computation time heavily increases with the number of design parameters in $\mathcal{P}$. Currently we have been able to solve design problems with up to 29 parameters by using distributed implementation of our algorithms. Indeed although we have not mentioned yet this point interval analysis algorithms are, by essence, appropriate for such a distributed implementation with, for example, a master computer managing the list of boxes and sending boxes to slave computers that perform a few iterations of the algorithm on the received box and send back the result (new boxes and solutions) to the master computer.

We consider a Gough platform with planar base and end-effector and similar legs. The 6 attachments points of the legs on the base are supposed to lie on a circle of radius R_1 with two adjacent points separated by an angle α (figure 3). The locations of the attachment points A_i on the base are fully defined if R_1, α are known. Similarly the locations of the B_i on the platform are fully defined by the radius r_1 of the platform and the angle β. The linear

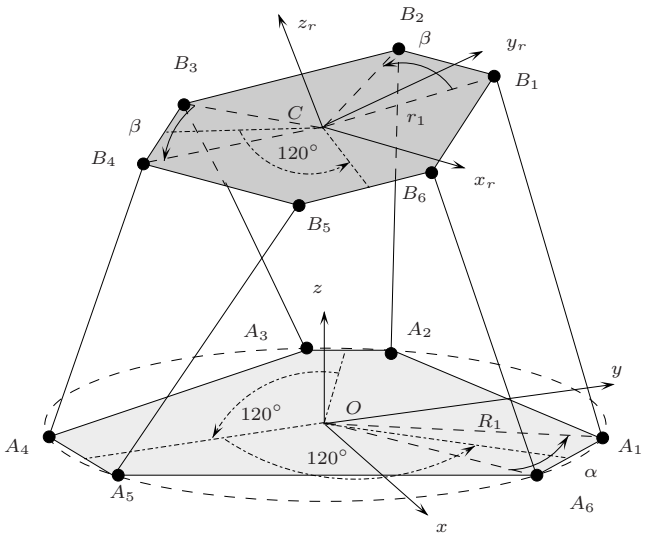

Fig. 3. The design parameters for a Gough platform

actuator in the leg has a stroke S and the minimal length of the leg is ρ_{min}. Hence the leg lengths ρ are constrained to lie in the range $[\rho_{min}, \rho_{min} + S]$. Our set of 5 design parameters $\mathcal{P}$ is defined as $R_1, \alpha, r_1, \beta, \rho_{min}, S$.

The requirements are that all poses of a given workspace should be reachable by the robot, that this workspace should be singularity-free (Fang and Merlet, 2005). Furthermore bounds for the sensor measurement errors $\Delta\rho$ are supposed to be known and their influence on the positioning errors Δq of the platform should not exceed a given threshold (note that these quantities are related by $\Delta\rho = J^{-1}(q)\Delta q$). Figure 4 shows a cross section in the α, β, R_1 space of the parameters space volume that is obtained as design solution.

7. Conclusion

In this paper we have shown that certification is an important issue in robotics, while uncertainties in such electromechanical system are unavoidable. A possible tool to obtain this certification and managing uncertainties is interval analysis. Some typical robotics problems have been presented but numerous other issues have also been addressed such as workspace analysis (Chablat, Wenger and Merlet, 2002), robots performance comparison (Chablat, Wenger and Merlet, 2004), calibration (Daney *et al.*, 2006) or robust control (Didrit, 1997).

Interval analysis may be used for small and medium size problems (although problems with up to 400 unknowns have been solved). One of the drawbacks of this

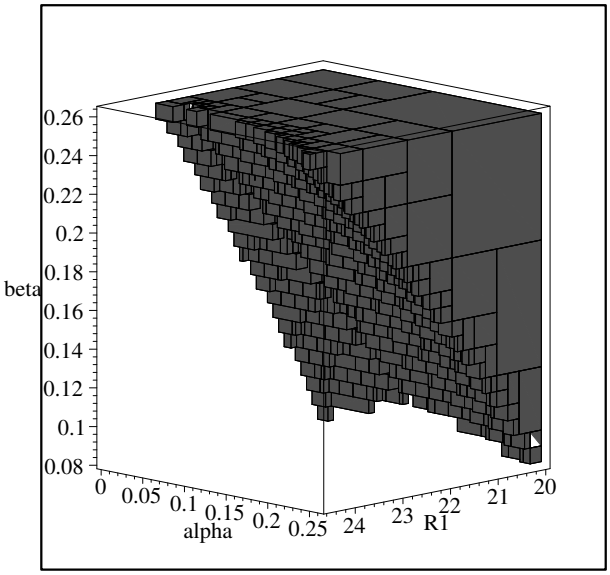

Fig. 4. A cross-section in the α, β, R_1 space of the design solution region.

method is that, although its basic principles are quite simple, an efficient implementation requires to *think* in term of interval analysis when formulating the problem, an extended knowledge of possible heuristics to be applied on the problem at hand and the availability of performing and complete interval analysis libraries. For the later point the interval analysis community must accept to make the important effort of providing unified libraries and to work on possible interfaces between these libraries and common scientific and engineering software such as `Maple`, `Scilab`.... This is indeed a very important issue as most end-users are not willing (or do not have the time) to learn a specific programming language to apply interval analysis just to solve a few steps of their whole engineering problems, that they have already formulated in one of the current engineering software.

References

Ashokaraj, I. et al. (July, 20-23, 2004). Sensor based robot localisation and navigation: Using interval analysis and extended Kalman filter., *5th Asian Control Conference*, Melbourne.

Carreras, C. and Walker, I. (2001). Interval methods for fault-tree analysis in robotics, *IEEE Trans. on Reliability* **50**(1): 3–11.

Chablat, D., Wenger, P. and Merlet, J.-P. (April, 1-4, 2004). A comparative study between two three-dof parallel kinematic machines using kinetostatic criteria and interval analysis, *11th IFToMM World Congress on the Theory of Machines and Mechanisms*, Tianjin, pp. 1209–1213.

Chablat, D., Wenger, P. and Merlet, J.-P. (June 29- July 2, 2002). Workspace analysis of the Orthoglide using interval analysis, *ARK*, Caldes de Malavalla, pp. 397–406.

Clerenti, A. et al. (September, 16-18, 2003). Imprecision and uncertainty quantification for the problem of mobile robot localization, *Performance Metrics for Intelligent Systems Workshop*, Gaithersburg.

Daney, D., Andreff, N., Chabert, G. and Papegay, Y. (August 2006). Interval method for calibration of parallel robots: a vision-based experimentation, *Mechanism and Machine Theory* **41**(8): 929–944.

Das, I. and Dennis, J. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problem, *Structural Optimization* **14**: 63–69.

Didrit, O. (June, 30, 1997). *Analyse par intervalles pour l'automatique; Résolution globale et garantie de problèmes non linéaires en robotique et en commande robuste*, PhD thesis, Université Paris XI Orsay, Paris.

Dietmaier, P. (June 29- July 4, 1998). The Stewart-Gough platform of general geometry can have 40 real postures, *ARK*, Strobl, pp. 7–16.

Fang, H. and Merlet, J.-P. (February 2005). Multi-criteria optimal design of parallel manipulators based on interval analysis, *Mechanism and Machine Theory* **40**(2): 151–171.

Gough, V. and Whitehall, S. (May 1962). Universal tire test machine, *Proceedings 9th Int. Technical Congress F.I.S.I.T.A.*, Vol. 117, London, pp. 117–135.

Hansen, E. (2004). *Global optimization using interval analysis*, Marcel Dekker.

Hubert, J. and Merlet, J.-P. (June, 23-26, 2008). Singularity analysis through static analysis, *ARK*, Batz/mer, pp. 13–20.

Innocenti, C. (June 2001). Forward kinematics in polynomial form of the general Stewart platform, *ASME J. of Mechanical Design* **123**(2): 254–260.

Jaulin, L., Kieffer, M., Didrit, O. and Walter, E. (2001). *Applied Interval Analysis*, Springer-Verlag.

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

Kieffer, M., Jaulin, L., Walter, E. and Meizel, D. (August 2000). Robust autonomous robot localization using interval analysis, *Reliable Computing* **6**(3): 337–362.

Kreinovich, V. (2000). Optimal finite characterization of linear problems with inexact data, *Technical Report CS-00-37*, University of Texas at El Paso.

Kreinovich, V., Lakeyev, A., Rohn, J. and Kahl, P. (1998). *Computational complexity and feasibility of data processing and interval computations*, Kluwer.

Lebbah, Y., Michel, C., Rueher, M., Merlet, J.-P. and Daney, D. (2004). Combining local consistencies with a new global filtering algorithm on linear relaxations, *SIAM J. of Numerical Analysis* .

Merlet, J.-P. (2004). Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis, *Int. J. of Robotics Research* **23**(3): 221–236.

Merlet, J.-P. (June, 14-16, 2000). ALIAS: an interval analysis based library for solving and analyzing system of equations, *SEA*, Toulouse.

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

Merlet, J.-P. and Donelan, P. (June, 26-29, 2006). On the regularity of the inverse jacobian of parallel robot, *ARK*, Ljubljana, pp. 41–48.

Moore, R. (1979). *Methods and Applications of Interval Analysis*, SIAM Studies in Applied Mathematics.

Neumaier, A. (1990). *Interval methods for systems of equations*, Cambridge University Press.

Neumaier, A. (2001). *Introduction to Numerical Analysis*, Cambridge Univ. Press.

Piazzi, A. and Visioli, A. (2000). Global minimum-jerk trajectory planning of robot manipulators, *Trans. on Industrial Electronics* **47**(1): 140–149.

Rao, R., Asaithambi, A. and Agrawal, S. (1998). Inverse kinematic solution of robot manipulators using interval analysis, *J. of Mechanical Design* **120**(1): 147–150.

Redon, S. et al. (2004). Fast continuous collision detection for articulated models, *9th ACM symposium on Solid modeling and applications*, Genoa, pp. 145–156.

Rex, G. and Rohn, J. (1998). Sufficient conditions for regularity and singularity of interval matrices, *SIAM Journal on Matrix Analysis and Applications* **20**(2): 437–445.

Ronga, F. and Vust, T. (1992). Stewart platforms without computer?, *Conf. Real Analytic and Algebraic Geometry*, Trento, pp. 197–212.

Rouillier, F. (1995). Real roots counting for some robotics problems, *in* B. R. J-P. Merlet (Ed.), *Computational Kinematics*, Kluwer, pp. 73–82.

Seignez, E. et al. (August, 2-6, 2005). Experimental vehicle localization by bounded-error state estimation using interval analysis, *IEEE/RJS IROS*, Edmonton.

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

Wampler, C. (April 1996). Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates, *Mechanism and Machine Theory* **31**(3): 331–337.

Wu, W. and Rao, S. (2004). Interval approach for the modeling of tolerances and clearances in mechanism analysis, *J. of Mechanical Design* **126**(4): 581–592.