Interval analysis and reliability in robotics

Jean-Pierre Merlet

INRIA, Sophia-Antipolis,
France
Email: Jean-Pierre.Merlet@sophia.innia.fr

Abstract: A robot is typical of systems that are inherently submitted to uncertainties although they should be highly reliable (i.e. for a robot used in surgical applications). The sources of uncertainties are the manufacturing tolerances of the mechanical parts constituting the robot which make the real robot always different from its theoretical model and control errors. We exhibit properties of a robot that are sensitive to the uncertainties and we present methods, using mainly interval analysis, that allow one to manage these uncertainties to ensure the reliability of the robot.

Keywords: interval analysis; robotics.

Reference to this paper should be made as follows: Merlet, J-P. (2009) 'Interval analysis and reliability in robotics', *Int. J. Reliability and Safety*, Vol. 3, Nos. 1/2/3, pp.104–130.

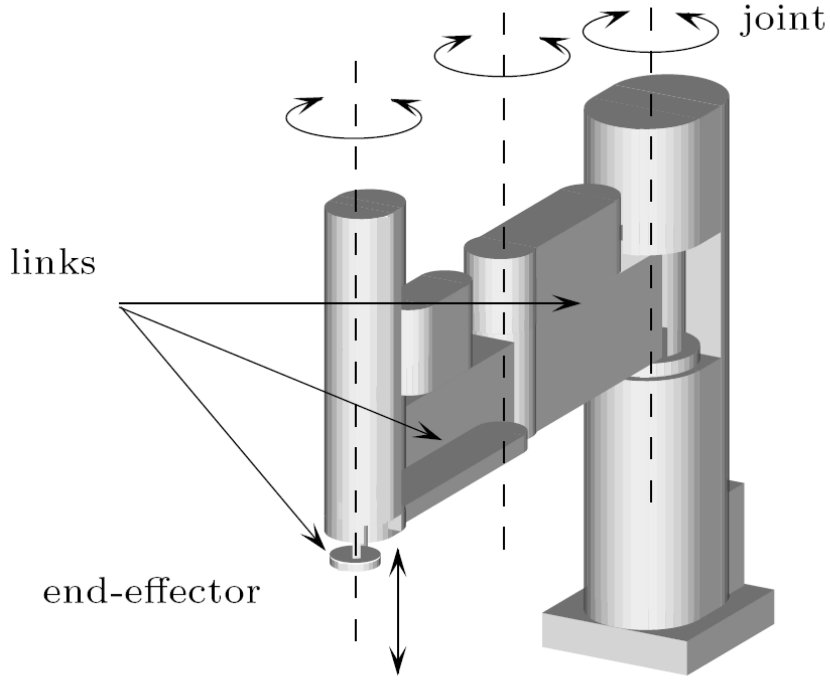
Biographical note: Jean-Pierre Merlet is a Senior Researcher at INRIA and was Head of the INRIA SAGA (1998) and COPRIN (2002) projects. He obtained his PhD from Paris VI University in 1986. He has published five books, approximately 150 conference papers and 40 journal papers in robotics, algebraic geometry, interval analysis, mechanism theory, geometry, control theory and mechanism history. He is currently chair of the French IFToMM Committee and Associate Editor of *Mechanism and Machine Theory* and *ASME Journal of Mechanisms and Robotics*.

1 Introduction

A robot is basically a mechanical system whose main task is to move an object or itself according to the decision of a high-level motion planner. Without losing to much generality we will restrict ourself in this paper to *industrial robots*. These types of robots have a *base* that is attached to the ground and an *end-effector* that is in charge of grasping and moving the objects. Hence the end-effector motion has to be controlled. An articulated mechanism connects the base and the end-effector. In the mechanism, we have *joints* (such as revolute or prismatic joints denoted respectively **R**, **P**), some of which are *actuated*, i.e. they are used to control the end-effector motion, while others are *passive* (i.e. they just follow the mechanism motion). A *pose* of the end-effector is a configuration in which all its six Degrees of Freedom (DoF) are fixed.

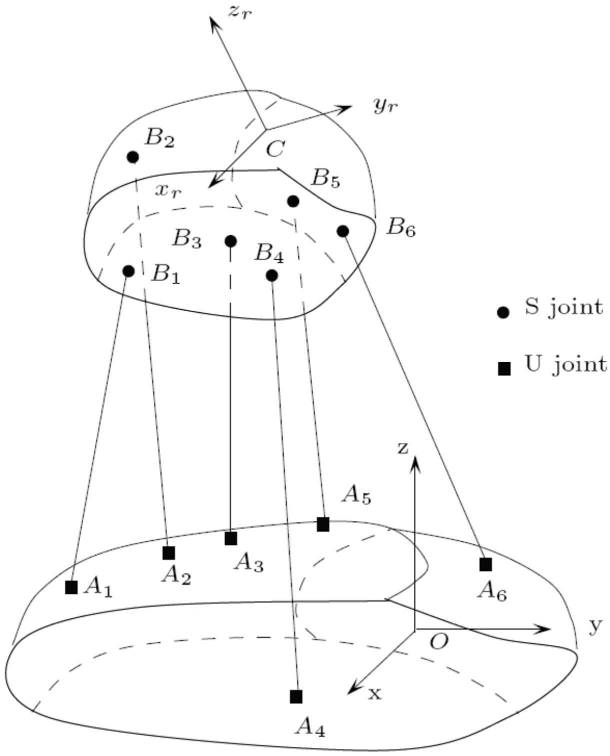
There are two main classes of industrial robots mechanical architecture: *serial* or *parallel*. In serial robot a succession of joint and rigid bodies leads from the base to the end-effector. Most industrial robots belong to this class, such as the *Scara* robot (Figure 1) which is constituted of the succession of *RRRP* joints, all actuated, that allow to control four DoF of the robot (the three translations along the reference axis and one rotation around the *z* axis). Another classical example of serial robot is the 6R robot constituted of a succession of six actuated revolute joints, that allows to control all six DoF of the end-effector. Most of six DoF industrial robot have such architecture with the peculiarity that the axis of the last 3R joints are concurrent in the same point, for a reason that will be explained later on.

Figure 1 The SCARA serial robot, a robot allowing the control of four degrees of freedom of the end-effector



The second class of architecture is the parallel robot in which different kinematics chains connect the base to the end-effector. The most famous parallel robots is the Gough platform in which six articulated legs connect the base to the platform (Figure 2). Controlling the length of the six legs allows one to control all six degrees of freedom of the end-effector. Gough platforms are widely used for flight simulators. Let us introduce classical notation in robotics. The vector **X** is constituted of a set of parameters that define the values of the n degrees of freedom of the end-effector. This vector is decomposed into the parameters **T**, **Γ** that describe, respectively, the translation and orientation of the end-effector. In the most general case, **X** is a six-dimensional vector, three of its components allowing to define the translation of the end-effector and the remaining three being angles allowing to define its orientation. The vectors $\mathbf{\Theta_a}$,$\mathbf{\Theta_p}$ define the value of respectively the actuated and passive joints of the robot (for example the rotation of a revolute joint).

Figure 2 A parallel robot: the Gough platform. By changing the lengths of the six legs we can control the pose of the upper body. S joint allows for any rotation around a given point while U joint allows for rotation around two mutually perpendicular axes intersecting at a point



As we will see many robotic problems may be safely solved by using methods based on interval analysis and we outline here the basic principle of this method (further details can be found in Moore, 1979; Neumaier, 1990; Hansen, 1992; Jaulin et al., 2001). Let **X** be a set of *n* unknowns that are supposed to be included in ranges that define an n-dimensional box *B0.* The problem to solve is to verify if a given property $\mathcal{P}$ is satisfied for some **X** in *B0.* We have a *decision operator D*(*B*) that takes as input a sub-box *B* of *B0* and returns 1 if the property is satisfied over *B*, –1 if it is not satisfied and 0 if the property cannot be ascertained. Most of the decision operators that we will use will be based on *interval arithmetics.* This arithmetic allows to substitute variables by intervals in an expression and to get an interval for the value of the expression that is *guaranteed* to include all possible values of the expression whatever are the values of the variables in their respective ranges. Consider for example the expression

$$x\cos(x) + y\sin(y)$$

and assume that *x* lies in the range [0,1] and *y* in the range [1,2]. The cosine of the range [0,1] is the range [cos(1),1] while the sine of the range [1,2] is [sin(1), 1]. Hence the expression can be written as [0,1] × [cos(1),1] + [1,2] × [sin(1), 1]. The first term is included in the range [0,1] while the second term lies in the range [sin(1), 2]. Consequently, the whole expression may be evaluated as [sin(1), 3].

Hence interval arithmetics allow one to determine lower and upper bounds for a given function, being given ranges for the unknowns. The range provided by interval arithmetic is called an *interval evaluation* of the function. Note that the interval evaluation usually overestimates the real values of the maximum and minimum of a given function. For instance in the previous example the interval evaluation of x cos(x) for x in [0,1] has been calculated as [0,1] through the result of the product of the two intervals [0,1], [cos(1), 1]. But clearly there is no x in [0,1] such that x cos(x) is equal to 1. Such overestimation is due to the multiple occurrences of the same variable in the expression. When calculating the interval evaluation of x cos(x) we proceed in the same way than for the interval evaluation of x cos(z). However, the amplitude of the overestimation will decrease with the width of the ranges and other methods may be used to calculate an interval evaluation with a lower overestimation.

An interesting property of interval arithmetics is that it can be implemented to take into account numerical round off errors due to the computers. A direct consequence is that the use of interval arithmetics allows to *certify* the decision even with respect to round-off errors. For example, it may occur that the property cannot be ascertained even for fixed values of the unknowns as numerical errors in the computer arithmetics do not allow to certify the verification of the property and this case will be detected. However, interval arithmetics has the drawback that its overestimation can lead to large computation time as numerous iterations may be necessary to ascertain the property.

In an interval analysis scheme sub-boxes of B_0 will be stored in a list L. Initially, L has a single element B_0 and sub-boxes will be added to the list during the algorithm: m will denote the number of elements in the list at each iteration of the process. A typical interval analysis algorithm uses an index i to indicate which box in L is processed, the index being initialised to 0.

1 if $i > m$, then exit

2 if $D(B_i) = -1$, then $i = i + 1$, go to step 1

3 if $D(B_i) = 1$, then store B_i as a solution, $i = i + 1$, go to step 1

4 if $D(B_i) = 0$, then bisect one of the range of the box and add the two resulting boxes in the list, $m = m + 2$, $i = i + 1$, go to 1.

We have described here only the very basic principle of the algorithm. Efficiency can be drastically improved, for example, by using a *filter* on the box B_i that allows to reduce its size or even discard the whole box (filters are usually specific of the solving problem). Example of such filters will be provided in some sections of this paper. Hence interval analysis cannot be considered as a 'black box' and requires some expertise to determine the appropriate formulation of the problem and the right combination of filters to get the best efficiency. Note also that the basic scheme of interval analysis is appropriate for a distributed implementation: a master programme manages the list of boxes and send the boxes to slave computers that will run a few iterations of the solving scheme and then returns the result to the master computer.

Interval analysis will be used in robotics because it allows to take into account the unavoidable uncertainties in the modelling of the robot or in its control without any hypothesis on their distributions (e.g. Gaussian or uniform), hence managing even the worst case.

Classical interval analysis notation will be used:

- the lower (upper) bound of a range x will be denoted $\underline{x}\,(\overline{x})$
- the *mid point* of a range x is the real $(\underline{x}+\overline{x})/2$
- the *width* of a range x is the real $\overline{x}-\underline{x}$
- the mid-point of a n-dimensional box is the point whose coordinates are the mid-point of the ranges that constitute the box.

Interval analysis has been used in the past to solve robotics problems such as robots navigation and localisation (Kieffer et al., 2000; Ashokaraj et al., 2004), reliability analysis (Carreras and Walker, 2001; Wu and Rao, 2004), collision avoidance (Fang and Merlet, 2005), robots synthesis (Tagawa et al., 2001; Lee et al., 2004) and kinematics (Rao et al., 1998). However, although the basic principles of interval analysis are simple, efficient implementation requires a high expertise level. We will now present examples of such implementation. For most of the problems presented in the next sections interval analysis is the only method that allows to solve *safely* the problem, i.e. to obtain a *guaranteed* result. The algorithms presented in the following sections have been implemented using our C++ library `ALIAS`[1] that uses the interval arithmetics `BIAS/PROFIL` and is interfaced with the symbolic computation software *Maple*.

2 Kinematics

A basic problem in robotics is to determine the relation between the actuated joint parameters and the pose of the end-effector. This is a crucial issue as the purpose of a robot is to allow to control the pose of the platform (i.e. the values of the elements of **X**) through the control of the actuated joints (i.e. the values of the elements of **Θa**). In the most general case, a geometrical analysis of the mechanism allows one to establish an implicit relationship

$$\mathbf{F}(\mathbf{X},\mathbf{\Theta_a},\mathbf{\Theta_p},\mathcal{P})=0 \tag{1}$$

where $\mathcal{P}$ denotes elements characterising the geometry of the robot as, e.g, location of the axis of revolute joints, distances between two joints. But this relationship should be simplified to generate more useful kinematics relations, namely the *inverse and direct kinematics.* The inverse kinematics ($\mathbf{F_i}$) gives directly the value of the actuated joints variables as functions of **X** while the purpose of the direct kinematics ($\mathbf{F_d}$) is to provide the value of **X** as functions of the actuated joints variables

$$\mathbf{\Theta_a}=\mathbf{F_i}(\mathbf{X},\mathcal{P}) \qquad \mathbf{X}=\mathbf{F_d}(\mathbf{\Theta_a},\mathcal{P}) \tag{2}$$

Note that in both cases, the passive joint variables have been eliminated to get a direct relation between the *input* (i.e. usually the actuated joints variables) and the *output* of the system (the location of the end-effector of the robot). The usefulness of the inverse kinematics is clear: as one want to control **X** (i.e. a desired value of **X** is known) it is necessary to determine what should be the orders given to the motors (i.e. the $\mathbf{\Theta_a}$) so that the end-effector reaches this pose. But the direct kinematics is also very useful as soon as one want not only to reach a given pose $\mathbf{X_1}$ from another pose $\mathbf{X_2}$, but also to do it by following a given trajectory (for example a line or a circle). The actuated joints variables $\mathbf{\Theta_a}$ are measured by sensors and there are always uncertainties in the control. Hence, the robot end-effector always deviates from its nominal trajectory and the direct kinematics is used to determine the amount of deviation, in order to correct it.

For serial robot, the direct kinematics problem is usually easy to solve: indeed, starting from the base, the first joint variable allows to determine the location of the second joint and so on until we reach the end-effector. But usually the inverse problem (which amounts to solve the system of direct kinematics equations) is difficult to solve. For example for the most general case of 6R robot, it is only relatively recently that it was shown that the problem has up to 16 solutions and that it was possible to exhibit an algorithm that is able to calculate them (Raghavan and Roth, 1990; Wampler and Morgan, 1991).

Reciprocally the inverse kinematics problem for parallel robots is usually simple: the locations of the extremities of the kinematics chain attached to the end-effector are known as soon as the location of the end-effector is given. It remains then to solve independently the inverse kinematics of each chain and these chains are usually designed so that it is a simple task. For example, for the Gough platform the actuated joint variables are the lengths of the legs, that can be calculated independently for each leg directly from **X**, the solution being unique (see Section 2.1). On the other hand, the direct kinematics is much more difficult and has usually multiple solutions. For the Gough platform it has been shown only recently that the problem has up to 40 real solutions (i.e. for a given set of leg lengths there may be 40 possible poses for the end-effector) (Husty, 1996; Dietmaier, 1998).

The inverse kinematics is basically used in a real-time context for the control of a robot. This is not a problem for parallel robots as they have usually a closed-form solution for this problem. But this will usually be problematic for serial robots such as the 6R. To simplify drastically the problem most industrial robot uses a simple trick: the axes of the three last R joints are concurrent at the same point *C*. **X** is then defined as the coordinates of *C* and the three angles represents the rotation around this point. In that case, the last three joint angles control only the rotation while the three first angles allows one to control the location of *C*: the problem has been decoupled, allowing for real-time computation.

Direct kinematics is also used for control purposes; this is not a problem for serial robot with their usually unique closed-form solution but is a difficulty for parallel robots. The direct kinematics problem has the following features:

1. it is crucial to determine the *current pose*, i.e. the pose of the robot when the sensor data were measured and not another pose that is a solution of the direct kinematics
2. the direct kinematics is solved at regular sampling time of the controller. Hence, as the velocity of the end-effector is limited, we can define a neighbourhood around the solution $\mathbf{X_{k-1}}$ obtained at time t_{k-1} that should include the solution at time t_k. If more than one solution is found in this neighbourhood, then the robot must be stopped as we are not able to determine the current pose
3. the neighbourhood including the solution is not avail able when the robot is started. Hence all solutions should be determined so that the end-user can determine which is the current pose among all the solution poses (indeed determining numerically the current pose among the solutions is still an open problem). However, in that case the real-time constraint can be relaxed.

As mentioned earlier, finding all solutions is difficult. Currently, the fastest approach is based on the use of Gröbner basis (Rouillier, 1995), but an approach based on interval analysis is almost as fast (Merlet, 2004). Current computation time ranges between a few seconds to one minute according to the geometry of the robot. Other methods have been proposed (mostly based on elimination or homotopy) but are not numerically safe (Mourrain, 1993; Wampler, 1996).

The real-time solution of the problem is based on item 3: the pose $\mathbf{X_{k-1}}$ is usually used as an initial guess for a Newton–Raphson scheme. But such scheme is not safe: the method may not converge or worst converge to a solution that is not the current pose (and this can happen even if the initial guess is very close to the current pose). Furthermore, this method does not allow to determine if there is more than one solution in the given neighbourhood, which is crucial for a safe use of the robot. In that case, interval analysis is very useful. Indeed the decision operator of the interval analysis scheme can use the Kantorovitch theorem (Tapia, 1971), that is presented hereafter.

Let **F** be a system of *n* equations in *n* unknowns:

$$\mathbf{F} = \left\{F_i\left(x_1,...,x_n\right) = 0, i \in \left[1,n\right]\right\}$$

Let x_0 be a point and U be a ball of radius $2B_0$ centred at x_0 i.e. the set of x such that $\|x - x_0\| \leq 2B_0\}$, the norm being $\|A\| = Max_i \sum_j |a_{ij}|$. Assume that x_0 is such that:

1 the Jacobian matrix of the system has an inverse $\mathbf{\Gamma}_0$ at x_0 such that $\|\mathbf{\Gamma}_0\| =\leq |A_0\|$

2 $\|\mathbf{\Gamma}_0\mathbf{F}(x_0)\| \leq 2B_0$

3 $\sum_{k=1}^{n} \left|\frac{\partial^2 F_i(x)}{\partial x_j \partial x_k}\right| \leq C$ for $i, j = 1,...,n$ and $x \in U$

4 the constants A_0, B_0, C satisfy $2nA_0B_0C \leq 1$

Then there is a unique solution of $\mathbf{F} = 0$ in U and Newton–Raphson iterative method used with x_0 as estimate of the solution will converge toward this solution. Conversely, let compute B_1 as $1/(2nA_0C)$ and assume that $B_1 \leq B_0$: then there is a unique solution in the box $[x_0 - 2B_1, x_0 + 2B_1]$.

Hence Kantorovitch theorem allows one to establish a ball that includes one, and only one, solution. It is even possible to enlarge this box by using an *inflation* method based on the H-matrix theory of Neumaier (1990, 2001). Assume that it has been determined that the ball $[x_0 - 2B_1, x_0 + 2B_1]$ includes one solution S of $\mathbf{F} = 0$. It can be shown that if the set $S_\mathbf{J}$ of Jacobian matrices of the system that are obtained in the ball $[x_0 - 2B_1 - \alpha, x_0 + 2B_1 + \alpha]$, where α is a positive constant, does not include any singular matrix, then this ball includes only S as solution of $\mathbf{F} = 0$. Evidently, calculating exactly the set $S_\mathbf{J}$ is difficult, but we can easily calculate a set of matrices that include $S_\mathbf{J}$. Indeed we can calculate an interval evaluation of each component of **J** over the ball $[x_0 - 2B_1 - \alpha, x_0 + 2B_1 + \alpha]$ and the resulting interval matrix defines a set of matrices that includes $S_\mathbf{J}$. We will mention in the *Singularity* Section 4 how to address the problem of checking the regularity of a set of matrices defined by an interval matrix but a possible method is to verify that the $n \times n$ interval matrix is *diagonally dominant*, i.e.

$$|J_{ii}| > \sum_{j=1, j\neq i}^{j=n} |J_{ij}|$$

for all i in $[1,n]$. This test can be used for an interval matrix and if it is diagonally dominant, then none of the matrices in the set is singular. In practice we start with a small α and increase it incrementally until the interval matrix is no more diagonally dominant.

The decision operator used for the direct kinematics solver returns –1 if the interval evaluation of at least one of the inverse kinematics equation does not include 0. If a solution is found through the Kantorovitch theorem the search domain is reduced to its complementary part with respect to the ball that contains the solution.

Hence interval analysis allows one either to safely calculate the single solution in the neighbourhood (i.e. to determine the current pose) or to determine that there is more than one solution in it.

2.1 An example

As an example we will consider the inverse kinematics of a Gough platform (Figure 2). We define a reference (O, **x**, **y**, **z**) that is attached to the base. A mobile frame (C, $\mathbf{x_r}$,$\mathbf{y_r}$,$\mathbf{z_r}$) is attached to the end-effector. The attachment points of the leg on the base (platform) are denoted by A_i (B_i). The pose of the end-effector is defined by the following parameters:

- x, y, z: the coordinates of the centre C of the end-effector in the reference frame
- ψ, θ, φ: three angles that describe the orientation of the end-effector. With these angles it is possible to calculate a rotation matrix **R** that transforms the components of a vector expressed in the mobile frame into its components in the reference frame. Typical of such angles are the Euler's angles with the rotation matrix

$$\mathbf{R} = \begin{pmatrix} c_\psi c_\phi - s_\psi c_\theta s_\phi & -c_\psi s_\phi - s_\psi c_\theta c_\phi & s_\psi s_\theta \\ s_\psi c_\phi + c_\psi c_\theta s_\phi & -s_\psi s_\phi + s_\psi c_\theta c_\phi & -c_\psi s_\theta \\ s_\theta s_\phi & s_\theta c_\phi & c_\theta \end{pmatrix}$$

where C_ψ, S_ψ denote the cosine and sine of the angle ψ.

The calculation of the leg length is identical for each leg so that we can drop the indices. Basically, this length is the Euclidean norm of the vector **AB**. This vector can be written as

$$\mathbf{AB} = \mathbf{AO} + \mathbf{OC} + \mathbf{CB} \qquad (3)$$

Being given the geometry of the robot the vector **AO** is known while the vector **OC** is an input of the problem. The vector **CB** can be written as $\mathbf{RCB_r}$ where $\mathbf{CB_r}$ denotes the coordinates of point B in the mobile frame and is given by the geometry of the robot, while the rotation matrix **R** is an input of the problem. Hence the three vectors on the right hand side of the equation (3) may be calculated from the geometry and from the pose **X** of the robot, which allow to determine the vector **AB** and then the leg length.

As for the direct kinematics we have to solve the system of six nonlinear equations $\rho_i^2 = \|\mathbf{A}_1\mathbf{B}_1\|^2$ derived from equation (3). Note, however, that other formulations can be used as well. For example, we can choose as unknowns the nine coordinates of three points B_i (e.g. the coordinates of $\mathbf{OB_1}$, $\mathbf{OB_2}$, $\mathbf{OB_3}$). For a given geometry we have

$\mathbf{OB}_j = \sum_{k=1}^{k=3} \alpha_k \, \mathbf{OB_k}$ for j in [4,6], where the α_k are constants which can be derived from the geometry of the end-effector. If d_{ij} denotes the known distance between the points B_i, B_j the constraint equations are

$$\begin{aligned} \rho_i^2 &= \|\mathbf{A_1B_1}\|^2 \; i \in [1,3] \\ d_{ij}^2 &= \left\|\mathbf{B_1O} + \mathbf{OB_j}\right\|^2 \end{aligned} \tag{4}$$

Equation (4) can be written as

$$(x_i - a_i)^2 + (y_i - b_i)^2 + (z_i - c_i)^2 = \rho_i^2 \tag{5}$$

where a_i,b_i,c_i, ρ_i are known constants and x_i,y_i, z_i are the unknown coordinates of the point B_i. The structure of this equation can be used to design a filter for an interval analysis method that will solve the direct kinematics. The equation can be written as

$$(x_i - a_i)^2 = \rho_i^2 - (y_i - b_i)^2 - (z_i - c_i)^2$$

For a given box of the interval analysis algorithm we can compute an interval evaluation [A_i, B_i] [C_i, D_i] of the left (right) hand side of the equation. If these intervals have no intersection (i.e. $A_i > D_i$ or $B_i < C_i$), then equation (5) has no solution and the box can be discarded. Now assume that the intersection [U_i, V_i] of the two intervals is strictly included in [A_i, B_i]. The value of xi is then included in the ranges $\left[\sqrt{U_i} + a_i, \sqrt{V_i} |+ a_i\right], \left[-\sqrt{V_i} + a_i, -\sqrt{U_i} + a_i\right]$, that may allow to update the range of this variable. Consider for example the following equation:

$$(x-1)^2 + (y-4)^2 + z^2 = 10$$

with x in [3,4], y,z in [1,2]. We get [A_i,B_i] = [4,9] and [C_i, D_i] = 10 − [4,9] − [1,4] = [−3,5]. The intersection of these ranges is [4,5] and x can be updated to $\left[3, \sqrt{5} + 1\right] \approx [3, 3.236]$.

A similar process can be used for the variable y_i, z_i and for each of the equations. For instance we write

$$z^2 = 10 - (x-1)^2 - (y-4)^2$$

with x in the range $\left[3, \sqrt{5} + 1\right], y, z \text{ in} [1,2]$. We get [$A_i$, B_i] = [1,4] and [C_i, D_i] = 10− [4,5] − [4,9] = [−4,2] with an intersection [1,2]. The range for z may thus be reduced to $\left[1, \sqrt{2}\right]$.

Such filter is called *local* as it uses only one of the equations of the system at the same time. There are also *global* filter that use either a subset or the whole set of equations.

The structure of the equation (4) leads also to interesting properties (Merlet, 2004):

- it allows to demonstrate a stronger version of the Kantorovitch theorem where the number n of equations can be substituted by 3. Hence the ball that includes a single solution of the system will be larger and as this ball is excluded from the search domain we get a faster algorithm
- the largest value of the constant α of the inflation method of Neumaier can be calculated formally instead of incrementally, this leading to a faster determination of the ball with a single solution.

3 Kinetics, accuracy and statics

The purpose of kinetics to study the relation between the joint velocities $\dot{\mathbf{\Theta}}$ and the end-effector velocities. The end-effector velocities can be decomposed into *translation velocities* $\dot{\mathbf{T}}$ and *angular velocities* $\mathbf{\Omega}$. Note that for robot involving rotations of the end-effector there are no orientation representations $\mathbf{\Gamma}$ whose derivatives are the angular velocities $\mathbf{\Omega}$. But we have $\mathbf{\Omega} = \mathbf{H}(\Gamma)\dot{\Gamma}$ where $\mathbf{H}$ is a matrix that is never singular (Angeles, 1997). Hence, we will use abusively the notation $\dot{\mathbf{X}}$ to denote the end-effector velocities vector. Derivation of the kinematics relations that can be written as $\mathbf{F}(\mathbf{X}, \mathbf{\Theta}) = 0$ allows one to establish the kinetic relation

$$\mathbf{A}\dot{\mathbf{X}} + \mathbf{B}\dot{\Theta} = 0 \tag{6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the partial derivatives matrices of $\mathbf{F}$. As for kinematics, we can define the inverse and direct kinetic relations as

$$\dot{\mathbf{X}} = \mathbf{J}\left(\mathbf{X}, \Theta, \mathcal{P}\right)\dot{\Theta} \quad \dot{\Theta} = \mathbf{J}^{-1}\left(\mathbf{X}, \mathbf{\Theta}, \mathcal{P}\right)\dot{\mathbf{X}} \tag{7}$$

where J is called the *Jacobian matrix* of the robot, which is a function of the pose of the robot and of its geometry. Note that these relations can also be used to perform a first order analysis of the accuracy of the robot. Indeed, the sensors that are used to measure the joint variables are affected by bounded measurement errors $\mathbf{\Delta\Theta}$. These errors will induce in turn a positioning error $\mathbf{\Delta X}$ for the end-effector as the control is using the measurement for adjusting the pose of the end-effector. The joint errors and the positioning errors are related by

$$\Delta\mathbf{X} = \mathbf{J}\left(\mathbf{X}, \Theta, \mathcal{P}\right)\Delta\Theta \tag{8}$$

In mechanics, there is a strong duality between kinetic and static analysis because of the virtual work theorem. If $\mathcal{F}$ denotes the forces/torques applied on the end-effector (that is usually called a *wrench)* and τ the forces or torques in the actuated joints, then we have

$$\tau = \mathbf{J}^{\mathrm{T}}\left(\mathbf{X}, \Theta, \mathcal{P}\right)\mathcal{F} \tag{9}$$

where $\mathbf{J}^{\mathbf{T}}$ denotes the transpose of J. As we can see kinetic and static analysis rely on the determination of the Jacobian matrix and on its inverse. For serial robots a closed-form for $\mathbf{J}$ is usually easy to obtain but a closed-form for its inverse can be difficult to get (or be too large to be useful), while for parallel robots a closed-form for $\mathbf{J}^{-1}$ is usually easy to obtain while a closed-form for $\mathbf{J}$ is difficult to get. For example the inverse

Jacobian of the Gough platform has as rows the normalised Plücker vector of the line associated to the legs of the robot:

$$J^{-1} = \left(\left(\frac{\mathbf{A_1B_1}}{\|\mathbf{A_1B_1}\|} \quad \frac{\mathbf{CB_1} \times \mathbf{A_1B_1}}{\|\mathbf{A_1B_1}\|} \right) \right) \tag{10}$$

Classical problems in robotics are related to equations (7–9). Let us assume that the end-effector will have to move within a known *workspace* W, i.e. that **X** is constrained to lie within a closed n-dimensional variety. For example the location of the centre of the end-effector will move within a sphere, while its rotation angles have to lie within some pre-defined ranges. The following problems have to be solved for any **X** in the workspace W.

- *type 1*: being given bounds on the actuated joints velocities, forces/torques, errors of the sensors, determine the maximal end-effector velocities, wrench, positioning errors
- *type 2*: being given bounds on the end-effector velocities, wrench, positioning errors, determine the maximal actuated joints velocities, forces/torques, sensors errors.

These problems are of the highest practical interest. A typical problem of type 1 can be illustrated for a medical robot. For such application the positioning accuracy of the robot, that will operate a real patient, is clearly of vital importance. When this type of robot is designed, joint sensors must be chosen so that the effects of the sensors measurement errors $\mathbf{\Delta\Theta^m}$ on the positioning errors of the robot are lower than a given threshold for any pose of the robot within the workspace in which it will have to operate: this can be called a *verification problem.* Hence we have to solve the following problem

$$\text{find Max } (\Delta\mathbf{X}) = \mathbf{J}(\mathbf{X}, \Theta, \mathcal{P})\Delta\Theta \quad \forall \mathbf{X} \in W \tag{11}$$

submitted to the constraint $|\Theta| \leq \Delta\Theta^m$. But a type 2 problem can also be derived from this equation. The cost of a sensor is highly dependent upon its accuracy and it is thus interesting to determine the maximal sensor error so that the maximal positioning errors of the end-effector are lower than given thresholds (this can be called the *design problem*). This can be formulated as follows:

$$\text{find Max } (\Delta\Theta) = \mathbf{J}^{-1}(\mathbf{X}, \Theta, \mathcal{P})\Delta\mathbf{X} \quad \forall \mathbf{X} \in W \tag{12}$$

submitted to the constraints $\Delta|\boldsymbol{X}_m| \leq \boldsymbol{X}_m^t$ for all components of $\mathbf{X}, \boldsymbol{X}_m^t$ being the threshold for the *m*-th component. Similar problems can be derived for the statics: verify that given actuated joints forces/torques allows to equilibrate a set of known wrenches or find the minimal joint forces/torques that will allow to equilibrate the wrenches for any pose of the end-effector within its workspace.

It can already be seen that for any type of robot one of these problems will be difficult to solve as either **J** or $\mathbf{J^{-1}}$ will not be known in closed-form. But any of these problems will be difficult as:

- we have to solve a *global* optimisation problem and the globality of the maximum must be guaranteed
- there are uncertainties in the geometry parameters $\mathcal{P}$ and the optimisation must take them into account.

But there is another approach to the problem. Indeed it is not strictly necessary to determine exactly the maximum of the optimisation problem. For example for the verification problem it is sufficient to show that the maximum of $\mathbf{\Delta X}$ is lower than the threshold. For the design problem not all values of the sensor errors are possible: indeed commercially available sensors provide only a finite set of possible accuracies, that can be ordered as $\{\mathbf{\Delta\Theta^0}, \mathbf{\Delta\Theta^1}, \ldots, \mathbf{\Delta\,\Theta}^{l}\}$. Hence determining that the maximal sensor error is greater than $\mathbf{\Delta\Theta}^k$ and lower than $\mathbf{\Delta\Theta}^{k+1}$ is sufficient to choose $\mathbf{\Delta\Theta}^k$ as a safe value for the sensor error.

If we look at equation (7) it appears that we have a linear relationship between the unknowns and that the optimisation problem (Kieffer et al., 2000; Jaulin et al., 2001) can be stated in one of the following two forms:

1. being given a vector **Y**, a known matrix **A** and $\mathbf{B} = \mathbf{A}(\mathbf{X}, \mathcal{P})\mathbf{Y}$, shows that the maximum, for all **X** in $\mathcal{W}$, of each component B_m of **B** is lower than a threshold B_m^t
2. being given a vector **B**, a known matrix **A** and **Y** such that $\mathbf{A}(\mathbf{X}, \mathcal{P})\mathbf{Y} = \mathbf{B}$, shows that the maximum, for all **X** in $\mathcal{W}$, of each component Y_m of **Y** is lower than a threshold Y_m^t

where **A** will be either the Jacobian matrix or its inverse according to the problem or the robot. For the first problem the use of interval analysis is appropriate. Indeed being given ranges for **X**, and even for the parameters in $\mathcal{P}$, interval arithmetics allows one to calculate an interval evaluation for $\mathbf{B} = \mathbf{A}(\mathbf{X}, \mathcal{P})\mathbf{Y}$. We can then design a decision operator that returns –1 if the lower bound of the interval evaluation is greater than B_m^t for all components of **B** or 1 if the upper bound is lower than B_m^t for at least one component of **B**. In the former case, the part of the workspace corresponding to the range for **X**, $\mathcal{P}$ will satisfy the constraints. In the later case, at least one constraint will be violated.

The second problem is more difficult as it is implicit, but interval analysis is also appropriate. Indeed, we have to consider the solutions of a set of linear systems and show that they are all included in a given bounding box. Assuming that **X**, $\mathcal{P}$ may be defined by ranges, we can consider that **A** is an *interval matrix* $\mathbf{A^i}$, i.e. a matrix with interval coefficients. Various methods for calculating a box that includes all solutions of all linear systems defined by any real matrix whose coefficients are included in the corresponding ranges of an interval matrix have been proposed (Neumaier, 1990; Kreinovich, 2000; Rohn, 2002). These methods should however be adapted as **A** is not only an interval matrix but has the stronger structure of a *parametric matrix.* Hence not all real matrices in the set defined by $\mathbf{A^i}$ will correspond to a Jacobian or inverse Jacobian. Consequently, the bounding box that is computed by classical methods, which is already larger than the optimal bounding box, will also be usually larger than the optimal bounding box that can be obtained for the set of Jacobian or inverse Jacobian matrices. Possible approaches to improve these methods is to take into account the derivatives of the coefficients with respect to **X**, $\mathcal{P}$ in the Gaussian elimination and to precondition **A** by pre- or post-multiplying it by a real matrix **K**. It can be shown that pre-conditioning may be very efficient for Jacobian matrices as soon as the multiplication **AK** (or **KA**) is first done symbolically in order to reduce the multiple occurrences of the unknowns (which is the main reason of the over-estimation of interval arithmetic), and then to plug-in the real coefficients of K to get the conditioned matrix.

4 Singularity

Singularity analysis is a crucial problem, especially for parallel robots. It can be intuitively explained by using the static equilibrium equation (9). Assume that the wrench $\mathcal{F}$ is given and that the joint forces/torques τ should be established. As this equation is linear each element of τ can be calculated as a ratio whose denominator is the determinant $|\mathbf{J}^{-\mathbf{T}}|$ of the transpose of the inverse Jacobian matrix. If this determinant is 0 and the numerator is not 0, then the joint force/torque will go to infinity, thereby leading to a breakdown of the robot. A pose **X** of the determinant is equal to 0 is called a *singular configuration* of the robot. A parallel robot usually exhibit such configuration (for example the reader can figure out that for a Gough platform whose base and platform are planar the configuration in which the base and platform lie in the same plane is a singular configuration) and some industrial prototypes have suffered from a spectacular breakdown when they have come close to a singular configuration.

Hence, clearly singularity should be avoided. This has to be done either for each trajectory performed by the robot or for the whole workspace $\mathcal{W}$ of the robot, as will be described in the following.

Usually the inverse Jacobian matrix is known in closed-form and it is maybe possible to calculate its determinant by using symbolic computation software (although it may be quite large). We have then to determine if this determinant may cancel within the workspace and the result should be guaranteed. There is only one method so far that has been proposed for this purpose (Merlet, 2007). An arbitrary point $\mathbf{X_1}$ is selected in $\mathcal{W}$ and the sign of its determinant is safely computed using an interval evaluation (without lack of generality we will assume that it is positive). The purpose of the algorithm is to determine if there can be a pose, or a set of poses, such that the determinant is negative. Indeed if this is the case, any path connecting one such pose and the pose $\mathbf{X_1}$ should cross a singular configuration. At the opposite if no such pose is found, then the workspace is singularity-free. We can thus use an interval analysis scheme with a decision operator for the property *singularity* that is based on the interval evaluation of the determinant. If its lower bound is positive, then the operators return –1. If the upper bound of the evaluation is negative, singularities are present in the workspace and the decision operators returns 1. If the lower bound is negative and the upper bound is positive the decision operator returns 0. This algorithm stops either when all boxes have been processed (in which case $\mathcal{W}$ is singularity-free) or as soon as a box with a negative determinant has been found.

A drawback of this algorithm is that it is difficult to take into account uncertainties in the geometry parameters $\mathcal{P}$ of the robot. Indeed if these uncertainties are added, the size of the closed-form of the determinant is so large that it cannot be calculated (this is, for example, the case for the Gough platform). However, having a closed-form for the determinant is not strictly necessary as an interval evaluation of the determinant may still be calculated by using classical linear algebra methods on the interval matrix that can be derived from the inverse Jacobian. A closed-form has however the advantage of providing a simplified form for the determinant with a reduced number of multiple occurrences of the unknowns and hence leads to a sharper interval evaluation of the determinant. Still, testing the regularity of an interval matrix is well known problem in interval analysis (Kreinovich et al., 1998; Kreinovich, 2000). One of the most efficient method has been proposed by Rohn (Rex and Rohn, 1998).

We define the set H as the set of all n-dimensional vector 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 $\mathbf{J}^{-1}$ at the i-th row and j-th column. Given two vectors **u**, **v** of H, we then define the set of matrices $\mathbf{A}^{\mathbf{uv}}$ whose elements $A^{\mathbf{uv}}_{ij}$ are

$$A^{\mathbf{uv}}_{ij} = \overline{a_{ij}} \text{ if } u_i v_j = -1, \underline{a_{ij}} \text{ if } u_i v_j = 1$$

These matrices have thus fixed numerical components corresponding to lower or upper bound of the interval J^{-1}_{ij}. There are 2^{2n-1} such matrices since $\mathbf{A}^{\mathbf{uv}} = \mathbf{A}^{-\mathbf{u},-\mathbf{v}}$. If the determinants of all these matrices have the same sign, then all the matrices $\mathbf{A}'$ whose components have a value within the interval evaluation of J^{-1}_{ij} are regular. Hence for the 6×6 $\mathbf{J}^{-1}$ of a Gough platform if the determinant of the 2048 matrices of $\mathbf{A}^{\mathbf{uv}}$ have the same sign, then all matrices in the set are regular.

But here again testing the regularity of the interval matrix that can be derived from the inverse Jacobian is only a sufficient condition, but not a necessary one as all real matrices derived from the interval matrix do not correspond to inverse Jacobian. Classically for parametric matrices **K**(**X**) Rohn test is not applied on the interval matrix directly, but on a *pre-conditioned* matrix **AK** where **A** is a real regular matrix. Usually this matrix is chosen as $\mathbf{K}^{-1}$(*Mid*(**X**)) where *Mid*(**X**) is the mid-point vector of the ranges for **X** (the purpose being to get a matrix **AK** that is closer to the identity matrix). But for a parametric matrix is better to first compute symbolically the pre-conditioned matrix and then to plug-in the value of the components of **A**. The most efficient methods seems to use an interval analysis scheme with a decision operator based on a symbolic pre-conditioning of the inverse Jacobian matrix and apply Rohn's regularity test on the pre-conditioned matrix. Consider for example the parametric matrix

$$\mathbf{K} = \begin{pmatrix} x & x \\ y & 2y \end{pmatrix} \tag{13}$$

where the unknowns x,y are supposed to lie in the range [1,2]. The determinant of the matrix being xy the set of parametric matrices does not include a singular matrix. The corresponding interval matrix is:

$$\mathbf{K}_I = \begin{pmatrix} [1,2] & [1,2] \\ [1,2] & [2,4] \end{pmatrix} \tag{14}$$

whose interval evaluation of the determinant is [–2,7]. We cannot expect the Rohn test to succeed as the singular matrix

$$\mathbf{K}_S = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} \tag{15}$$

is included in $\mathbf{K}_\mathrm{I}$. If we use a numerical pre-conditioning we get

$$\mathbf{A} = \begin{pmatrix} 4/3 & -2/3 \\ -2/3 & 2/3 \end{pmatrix} \tag{16}$$

and a pre-conditioned matrix

$$\mathbf{AK}_I = \begin{pmatrix} [0,2] & [-4/3,4/3] \\ [-2/3,2/3] & [0.2] \end{pmatrix} \tag{17}$$

The interval evaluation of the determinant of $\mathbf{AK}_I$ is [−8/9, 44/9], that does not allow to state that **K** is regular. Assume now that we assume that the components of **A** are a_{ij}. The symbolic pre-conditioned matrix $\mathbf{S}_s$ is

$$\mathbf{S}_S = \begin{pmatrix} x(a_{11}+a_{21}) & x(a_{12}+a_{22}) \\ y(a_{11}+2a_{21}) & y(a_{12}+a_{22}) \end{pmatrix} \tag{18}$$

If we now plug-in the numerical values of the a_{ij} [equation (16)] in this matrix we get

$$\mathbf{S}_K = \begin{pmatrix} 2x/3 & 0 \\ 0 & 2y/3 \end{pmatrix} \tag{19}$$

For the range [1,2] we get the matrix

$$\mathbf{S}_K = \begin{pmatrix} [2/3,4/3] & 0 \\ 0 & [2/3,4/3] \end{pmatrix} \tag{20}$$

and the interval evaluation of the determinant of this matrix is [4/3,16/3] allowing to state that **K** is never singular. Another method proposed by Popova for interval parametric matrices may also be useful in some cases (Popova, 2004). Combining all these methods has allowed to test the presence of singularity within the workspace of robots with a geometry that takes into account the manufacturing tolerances in a computation time that ranges from a few seconds to 10 minutes.

5 Trajectory verification

A robot is submitted to various mechanical constraints that does not allow it to perform arbitrary trajectories. For example the leg of a Gough platform have a minimum and maximum length and the mechanical joints on the base and platform allow only for a limited rotation of the legs, while singularity should be avoided. Typical mechanical constraints can be defined as

$$\mathbf{F}(\mathbf{X}) \le 0 \quad \mathbf{G}(\mathbf{X}) \ne 0 \tag{21}$$

We will see in the next section that we can deal with these constraints at the design stage (provided that a desired workspace $\mathcal{W}$ is defined) to ensure that the constraints are satisfied whatever is the pose of the robot within its workspace. But this is not sufficient as in some cases the robot has to perform trajectories that lie outside the prescribed workspace. Hence the software allowing to verify the constraints for an arbitrary trajectory is still necessary. Here again uncertainties should be taken into account. They can occur at two different levels:

1 *modelling*: the geometry of the real robot differs from its theoretical model that is used to calculate F, G

2 *control*: the trajectory followed by the robot will not be exactly the desired one as

- the theoretical model is used by the control law
- there are measurement errors in the internal sensors that are used to control the trajectory
- control errors will always occur.

A generic motion verifier can be designed as soon as the trajectory can be described as a time function. Without lack of generality we may assume that each component X_i of **X** can be written as an analytical function of the time T, which lie in the range [0,1]. As soon as a parser allows to calculate an interval evaluation of the functions **F**, **G** with respect to a time range we can always design a generic decision operator that checks if these constraints are verified for the whole interval [0,1] [such a parser has been proposed by Merlet (2001)]. Modelling uncertainties can be introduced as ranges in **F**, **G** while control errors, that can reasonably be bounded, are introduced as uncertainty ranges in the component of **X**. Hence if the trajectory is successfully checked, then we can ensure that the trajectory is safe, even in the worst case. Trials have shown that such a generic motion verifier can check realistic constraints in a few seconds, i.e. almost in real time.

But this approach can also be used to *optimise* the trajectory as will be shown in the following example.

5.1 Verification: an example

Assume that a Gough platform has to follow a circular trajectory with radius R in a horizontal plane at altitude $z = z_c$ with a constant orientation (the rotation matrix is the identity matrix). The trajectory can be defined as

$$
\begin{aligned}
x &= R\cos(2\pi T) + \Delta x \\
y &= R\sin(2\pi T) + \Delta y \\
z &= z_c + \Delta z
\end{aligned}
$$

where Δx, y, z represent the control errors (for simplicity reasons we will assume that there are no errors on the orientation but they can be introduced as well). The square of the leg lengths can be computed as

$$
\begin{aligned}
\rho^2 = &\left(R\cos\left(2\pi T\right) + \Delta x - x_a + x_b - \Delta x_a + \Delta x_b\right)^2 + \\
&\left(R\sin\left(2\pi T\right) + \Delta y + y_a + y_b - \Delta y_a + \Delta y_b\right)^2 + \\
&\left(z + \Delta z - z_a + z_b - \Delta z_a + \Delta z_b\right)^2
\end{aligned}
\tag{22}
$$

where the coordinates with the a (b) subscript represent the coordinates of A_i (B_i) and the Δ the modelling errors. The constraints that should be satisfied are

$$\rho_{min}^2 \le \rho^2 \le \rho_{max}^2$$

for all legs and for any T in the range [0,1]. Equation (22) can be written as the sum of the square of three terms U_j, V_j, W_j and one of the constraint can be written as

$$U_j^2 + V_j^2 + W_j^2 \le \rho_{max}^2$$

The structure of this inequality allows to design a filter for the interval analysis algorithm. Assume that there is a T such that the maximal leg length constraint is violated. The inequality can be written for example as

$$U_j^2 > \rho_{max}^2 - V_j^2 - W_j^2$$

Without going into the details it can be seen that the calculation of the interval evaluation of the right hand-side term can be used to eventually modify the interval evaluation of U_j^2. In turn this new interval evaluation can be used to update the range of the variable that appears in U_j: we can thus create a filter that will speed up the processing.

5.2 Optimisation: an example

Consider for example a Gough platform that is used as a milling machine (see Figure 7). Although the robot has six degrees of freedom only five are necessary for the task at hand as the spindle provides the rotation around the platform normal (which corresponds to the angle ϕ of the Euler angles). The value of ϕ is hence free and can be used to optimise the trajectory. First of all we have to ensure that the trajectory is *feasible*, i.e. the constraints on the leg lengths are respected over the whole trajectory. If we assume that the trajectory is defined through an analytical time functions for the pose parameters, then the leg lengths can be established as function of the time T and of ϕ. For a given time the square of a leg length can be written as $A \cos \phi + B \sin \phi + C$ where A, B, C are variables that depend only upon the geometry of the robot and upon the time. For a given leg and a given time the equations $\rho_i^2 = \rho_{min}^2, \rho_i^2 = \rho_{max}^2$ have generally 2 solutions in ϕ. Hence there will be one or two intervals for ϕ such that $\rho_{min}^2 \le \rho_i^2 \le \rho_{max}^2$.

If there are uncertainties in the geometrical model of the robot we are still able to determine ranges for ϕ so that the leg lengths constraints will be satisfied. We may then consider that the trajectory should be singularity-free. At a given time, the determinant of the inverse Jacobian is a polynomial P in $\sin \phi$, $\cos \phi$

$$P = r_{jk} \sin^j \phi \cos^k \phi$$

If we impose that the absolute value of P should be greater than a given threshold ε_s, then the structure of P is such that there will be in general ranges for ϕ such that P is greater than ε_s, even if there are uncertainties in the coefficients r_{jk}.

At this point we have still some freedom for the choice of ϕ for a given time, T. In the considered application, we can assume that the wrench applied by the tool on the robot is constant. A possible optimisation choice is to minimise the maximal absolute value of the actuator forces τ, such choice allowing to reduce the wear on the actuator. The purpose of the algorithm is to produce a time-law for ϕ as a set of values $\phi(0)$... $\phi(\Delta t)$, $\phi(2\Delta t)$,... $\phi(1)$ which can be used to control the machine. An initial value of Δt is provided by the user but will be adjusted by the algorithm if necessary.

A mix of the decision operators presented in the previous sections allows to determine ranges for ϕ such that in a given time range [T, $T + \Delta t$] the leg lengths constraints are satisfied and the trajectory is singularity-free. If such range cannot be found we divide Δt by 2 until

- at least one range is found or
- the algorithm establishes that the leg lengths constraints will be violated whatever the value of T or
- the value of the determinant will be lower than ε_s for some T in the range.

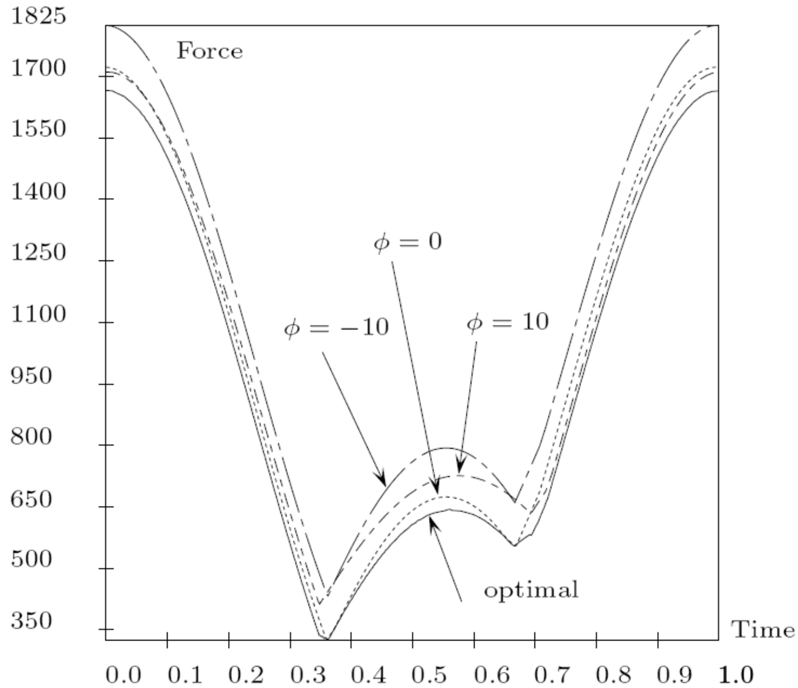
If we assume that ranges for ϕ have been found for a given time range [T_1, T_2] we will use another interval analysis algorithm to determine what is the value of ϕ that leads to τ_m, the minimal maximal absolute values for all the forces τ. However, being given the uncertainties in the task τ_m cannot be established as a real value but as a range. As usual we will consider the worst case and try to minimise $\overline{\tau_m}$. Still it does not make sense to calculate *exactly* the minimal value τ_m of $\overline{\tau_m}$ SO we will stop the calculation as soon as we can ascertain that we have determined a value of ϕ such that $\overline{\tau_m} - \tau_M \leq \alpha$, where α is a threshold that is fixed by the end-user.

Thus we have to solve a global optimisation problem and classical methods of interval analysis can be used for that purpose (Hansen, 1992). In our case we have to optimise a function of ϕ that can be derived from equation (9), while the parameter T lies in a given range. Determining exactly the value of $\overline{\tau_m}$ is an optimisation problem of type (Jaulin et al., 2001) and therefore relatively difficult as we do not have an explicit formulation of the function. But we can still use an interval analysis scheme with a decision operator that solve the linear interval system (i.e. determine a range that includes $[\underline{\tau_m}, \overline{\tau_m}]$) and uses the upper bound of this range as possible value for $\overline{\tau_m}$. A current value τ_m of $\overline{\tau_m}$ is maintained all along the algorithm. The decision operator returns

- -1 if $|\tau_m - \underline{\tau_m}| \leq \alpha$ or if $\underline{\tau_m} > \tau_M$
- -1 if $\overline{\tau_M} - \underline{\tau_m}$ is lower than α . In that case, we set all interval variables to the mid-point of their ranges, solve numerically the linear system (9) and use the result to eventually update τ_M
- 0 otherwise

We have implemented this approach in a generic way (Merlet, 2000) in the sense that the trajectory may be described with arbitrary analytical functions without having to change the kernel of the algorithm. For that purpose the parser presented in the previous section allows to calculate the necessary interval evaluations. Figure 3 presents a typical result of the algorithm for a circular trajectory. On these plots are presented the maximal absolute values of the joint forces for various fixed values for ϕ together with the maximal absolute values of the joint forces obtained with our method. Note that even for this simple trajectory the optimisation method allows for a significant decrease of the maximal forces.

Figure 3 Maximal absolute values of the joint forces for various fixed value for ϕ and for our optimisation method



6 Appropriate design

The mechanical system that constitutes the basis of a robot should be designed so that a set of basic performance requirements $\{\mathcal{R}_1,\ldots, \mathcal{R}_m\}$ are satisfied, whatever is the pose of the robot in the desired workspace. For that purpose we have to choose theoretical values for the design parameters $\mathcal{P}$, keeping in mind that the values of the parameters for the real robot will always differ from their theoretical values. A classical approach to satisfy the requirements is the *trial-and-error* method that is based on a simulation software that allows to establish the performances of a given geometry. But this approach is difficult to use in practice as the cross-coupling between the influences of the various parameters is difficult to manage by hand as soon as the number of geometry parameters exceeds a very small number. The classical academic way to deal with this problem is to use an *optimal design* methodology based on the *cost-function* approach (Erdman, 1993). In this approach an index I_k in the range [0,1] is associated to each performance requirement R_k, the value of this index increasing with the level of the deviation of the performance with respect to the requirement. These indices are clearly functions of the design parameters $\mathcal{P}$ and a cost-function $\mathcal{C}$ is defined as

$$\mathcal{C} = \sum_{k=1}^{k=m} w_k I_k(\mathcal{P}) \tag{23}$$

where w_k are weights. Then a numerical optimisation procedure is used to find the value of $\mathcal{P}$ that minimises the cost-function. There are numerous drawbacks to this approach:

- the definition of the indices I_k are not easy for some performance requirement (e.g. the shape of the workspace of the robot)
- the use of a numerical optimisation approach imposes that the computation time for the calculation of the indices should be low. However, we have seen in the previous sections that performance requirements such as accuracy or singularity detection require a significant computation time to be safely checked
- satisfaction of *imperative* requirements (e.g. requirements that *must* be satisfied by the robot) is difficult to strictly ensure
- if there are antagonistic performance requirements (typically size of the workspace versus accuracy) the cost-function provides only a compromise between the requirements that is deeply influenced by the choice of the weights (Das and Dennis, 1997). Furthermore there is neither an intuitive choice for these weights nor a general method to find appropriate values for them if the design solution is not satisfactory
- uncertainties are difficult to take into account

Another approach is based on our practical experience of industrial robot design. The point is that in practice we have to deal with a list of requirements, some of them being more important than the others, but none having to be optimised *per se.* The objective becomes more to satisfy *all* of the requirements which led us to an *appropriate design* methodology.

If $\mathcal{P}$ is an m-dimensional vector we can define a *parameter space* in which each dimension is associated to one of the design parameters. One point in this space represents a unique robot design. Then it must be noted that the design parameters of a robot are usually bounded i.e. that only design solutions within some known ranges should be found. Hence values for the design parameters should be found within an m-dimensional box of the parameter space.

The idea of the appropriate design is simple. We will consider in turn each of the requirements $\mathcal{R}_i$ and assume that we are able to determine the region $\mathcal{Z}_i$ of the parameter space that defines *all* design parameter values for which the requirement $\mathcal{R}_i$ is satisfied.

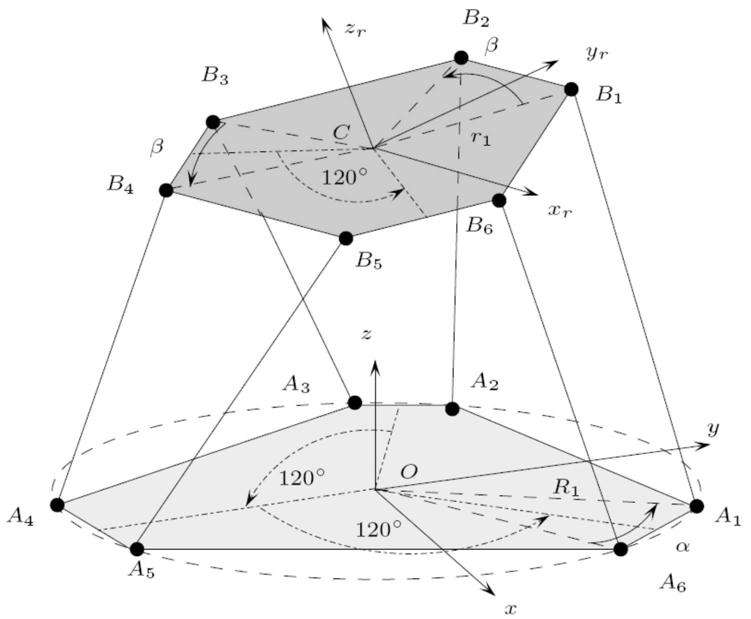
A design that will satisfy *all* requirements has therefore a representative point in the parameter space that belongs to all $\mathcal{Z}_i$, hence to their intersection. If we assume that we can calculate the regions $\mathcal{Z}_i$ and then their intersection, then we will get *all* the design solutions. In practice, however, we cannot propose to an end-user an infinite set of design solutions. Thus the intersection will be sampled to provide a finite set of design solutions that are representative of various compromises between the different performances (including performances that may have not been included initially in the desired requirements).

But the principle of this design methodology may seem to be quite theoretical. Indeed computing the regions is clearly quite difficult in most cases and calculating the intersection is also a difficult step. But here uncertainties will play, for once, a positive role. Indeed computing *exactly* the regions for the requirements is not necessary because points of the parameter space that belong to a $\mathcal{Z}_i$ that are close to its border are only theoretical solutions: if they are chosen as nominal values for the design parameters, their values for the real robot can lead to a representative point in the parameter space that does not belong to $\mathcal{Z}_i$ and the requirement $\mathcal{R}_i$ will not be satisfied by the robot. Hence only an approximation of the $\mathcal{Z}_i$ is necessary. To simplify further the calculation of the $\mathcal{Z}_i$ we may use a *relaxed* version of the requirements provided that the full version of the requirements can be checked for the design solution. In that context we will see on a realistic example that interval analysis is an appropriate tool.

6.1 An example

We consider a Gough platform with planar base and platform. The six 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 4). 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 actuator in the leg

Figure 4 The design parameters for a Gough platform



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

The requirement is that for any pose (x, y, z, ψ, θ, ϕ) such that $x \in \left[\underline{x},\overline{x}\right] y \in \left[\underline{y},\overline{y}\right] z \in \left[\underline{z},\overline{z}\right] \psi \in \left[\underline{\psi},\overline{\psi}\right] \theta \in \left[\underline{\theta},\overline{\theta}\right] \phi \in \left[\underline{\phi},\overline{\phi}\right]$, where the underlined and overlined quantities are known constants, the six leg lengths belong to the range $[\rho_{min}, \rho_{min} + S]$. In other words, we have defined a six-dimensional workspace $\mathcal{W}$ (which is a hyper-cube) for the robot and we require that any pose of $\mathcal{W}$ satisfies the constraint on the leg lengths.

We will relax somewhat the requirement for the design process by imposing that the leg lengths constraints are satisfied for the 64 corners {$\mathbf{X}_1$,..., $\mathbf{X}_{64}$} of the hyper-cube $\mathcal{W}$. As seen previously the square of the leg length ρ can be computed as an analytical function F(X, $\mathcal{P}$) of the pose **X** and of the design parameter $\mathcal{P}$. We will denote by $F_i = \left[\underline{F}_i, \overline{F}_i\right]$ the interval evaluation of the length ρ_i of the *i*-th leg.

All the design parameters may clearly be bounded. All of them should be positive, constraint on the overall size of the robot will give an upper bound for R_1 while we should have $r_1 < R_1$ and symmetry constraints impose an upper bound for α, β. We will use an interval analysis algorithm applied on the set of design parameters. An underlined or overlined parameter will indicate lower or upper bound for the parameters in the current box that is processed in the algorithm. The decision operator of this interval analysis algorithm is defined as follows:

1 returns −1 if $\overline{\mathcal{P}}_i - \underline{\mathcal{P}}_i < \varepsilon_i$ where ε_i are the manufacturing tolerances associated to the design parameter $\mathcal{P}_i$

2 returns 1 if $\underline{F}_i \geq \overline{\rho_{min}}$ and $\underline{F}_i \leq \overline{\rho_{min}} + \underline{S}$ for all $i \in [1,6]$ and all poses **X** in {$\mathbf{X}_1$,..., $\mathbf{X}_{64}$}

3 returns −1 if $\underline{F}_i > \overline{\rho_{min}} + \overline{S}$ or $\overline{F}_i < \underline{\rho_{min}}$ *in* for at least one *i* in [1,6] and one pose in {$\mathbf{X}_1$,... , $\mathbf{X}_{64}$}

4 returns 0 otherwise

Case 2 indicates that for all the 64 poses the constraints on the six leg lengths are satisfied, whatever the values of the design parameters are in their current range: this is clearly a design solution. This is even a robust design solution with respect to uncertainties: if the current parameter range for $\mathcal{P}$ is $\mathcal{P}_i^c = \left[\underline{\mathcal{P}}_i, \overline{\mathcal{P}}_i\right]$, then we can choose as nominal value for $\mathcal{P}_i$ any value in $\left[\underline{\mathcal{P}}_i + \varepsilon_i/2, \overline{\mathcal{P}}_i - \varepsilon_i/2\right]$ to ensure that for the real robot $\mathcal{P}_i$ lie in the range $\mathcal{P}_i^c$ and hence that it will satisfy the leg length constraints. On the contrary case 3 indicates that one leg constraint is violated whatever the design parameters values are for their current robot: no design solution can be found in the current box. Case 1 is the place where uncertainties are taken into account. There may be a theoretical design solution in the current box but even if one of those solutions was found we cannot ensure that the representative point of the real robot will still lie in the current box, and hence that the leg constraints will be satisfied. The boxes eliminated at step 1 are clearly on the border of the region $\mathcal{Z}_i$ and the result provided by the algorithm constitutes an approximation of this region.

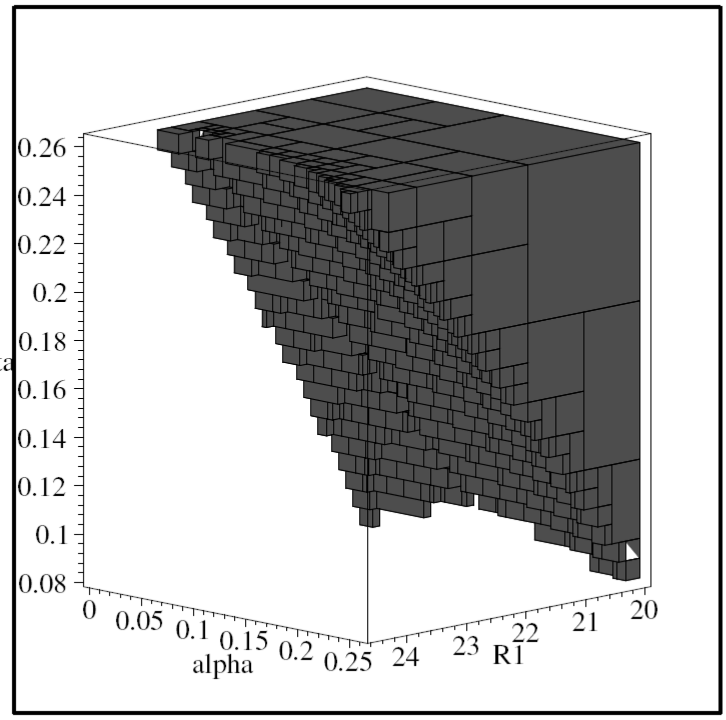
This algorithm has the following properties:

- *easy intersection calculation*: the result of the algorithm is a set of six-dimensional boxes. If all regions $\mathcal{Z}_i$ are computed with a similar algorithm the intersection of the obtained regions is simple to calculate. Furthermore the region $\mathcal{Z}_i$ can be used as *input* for the calculation of the region $\mathcal{Z}_{i+1}$, so that the region obtained for the requirement $\mathcal{R}_m$ will be directly the intersection of all $\mathcal{Z}_i$

- *quality of the approximation*: indices may be defined to qualify the quality of the calculation of $\mathcal{Z}_i$. Let $\boldsymbol{V}_{in}, \boldsymbol{V}_{ne}, \boldsymbol{V}_{out}$ be respectively the total volume of the approximation, of the boxes that are neglected in case 1 of the decision operator and of the boxes that are discarded (case 3). A relative quality index in the range [0,1] can be defined as $\boldsymbol{V}_{in}/(\boldsymbol{V}_{in}+\boldsymbol{V}_{ne})$ (the closer the index is to 1, the better the approximation). An absolute quality index, also in the range [0,1], can be defined as $(\boldsymbol{V}_{in} + \boldsymbol{V}_{out})/(\boldsymbol{V}_{in} + \boldsymbol{V}_{ne} + \boldsymbol{V}_{out})$.
- *incremental calculation of $\mathcal{Z}_i$*: instead of using the manufacturing tolerances for $\in$ we can start a run of the algorithm with larger values and store the neglected boxes of case 1 in a special file. The quality of the approximation (whose total volume will be $\boldsymbol{V}_{in}^1$) for this run can be calculated using the previous indices. If it is not satisfactory we may restart the algorithm with a lower $\in$ and with as input the boxes that have been neglected at the previous run. The boxes that will be obtained during this run have a total volume $\boldsymbol{V}_{in}^2$ and will be added to the boxes obtained at the first run, leading to an approximation with a total volume $\boldsymbol{V}_{in}^1 + \boldsymbol{V}_{in}^2$ while the total volume of the neglected boxes will be $\boldsymbol{V}_{ne}^2$. Hence we can incrementally improve the quality of the approximation until it is satisfactory, thereby reducing the computation time.

Figure 5 shows a cross section in the α, β, R_1 space of the volume that is obtained.

Figure 5 A cross-section in the α,β,R_1 space of *Zi.* A point in this space represents a robot geometry and the workspace of this robot is guaranteed to include a set of 64 pre-defined poses of the end-effector



We have designed similar algorithms for accuracy and static analysis and this has allowed us to design successfully robots for various applications. For example a precise positioner with a measured absolute accuracy better than 1 μm for a load of 2.5 tons has been designed for the European Synchrotron Radiation Facility (ESRF) (Figure 6). We have also collaborated with the SME constructions Měcaniques des Vosges for the design of a high speed milling machine (Figure 7). The interest of parallel robots for such application is the high rigidity and high velocities. A similar methodology has also been used for the design of a deployable space telescope for Alcatel (Fang et al., 2005) and for a surgical micro-robot (Figure 8).

Figure 6 The positioner for the European Synchrotron Radiation Facility (see online version for colours)

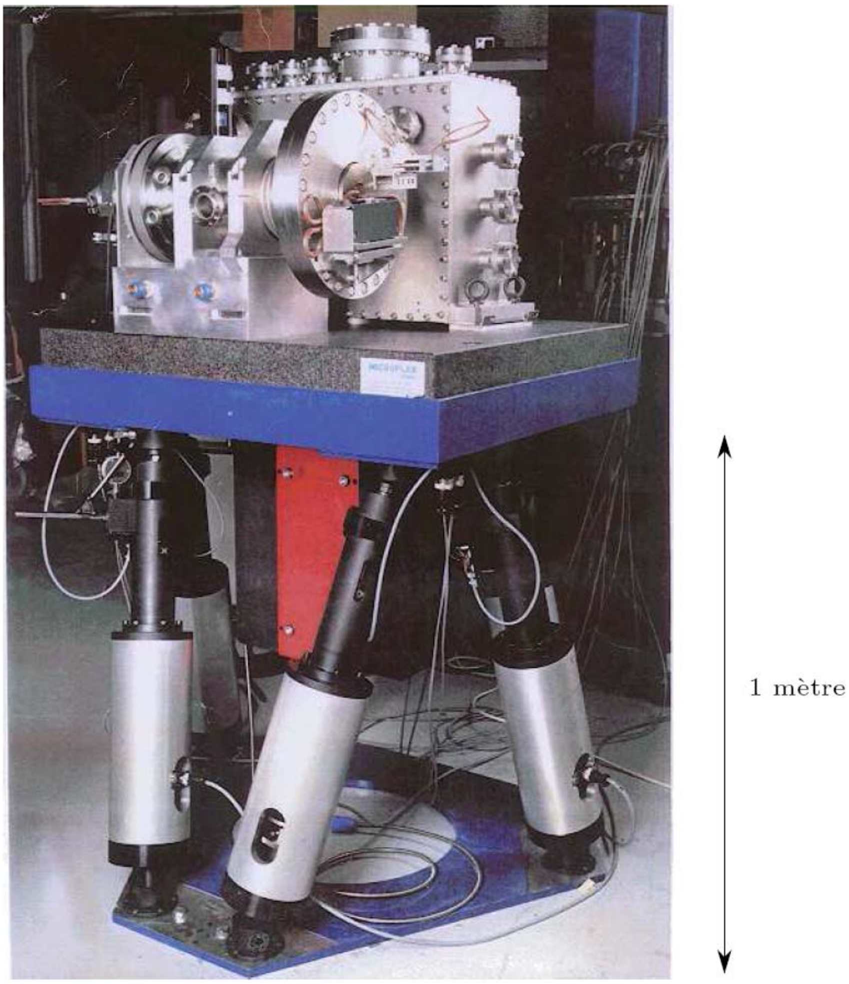


Figure 7 The high-speed milling machine of CMW (see online version for colours)

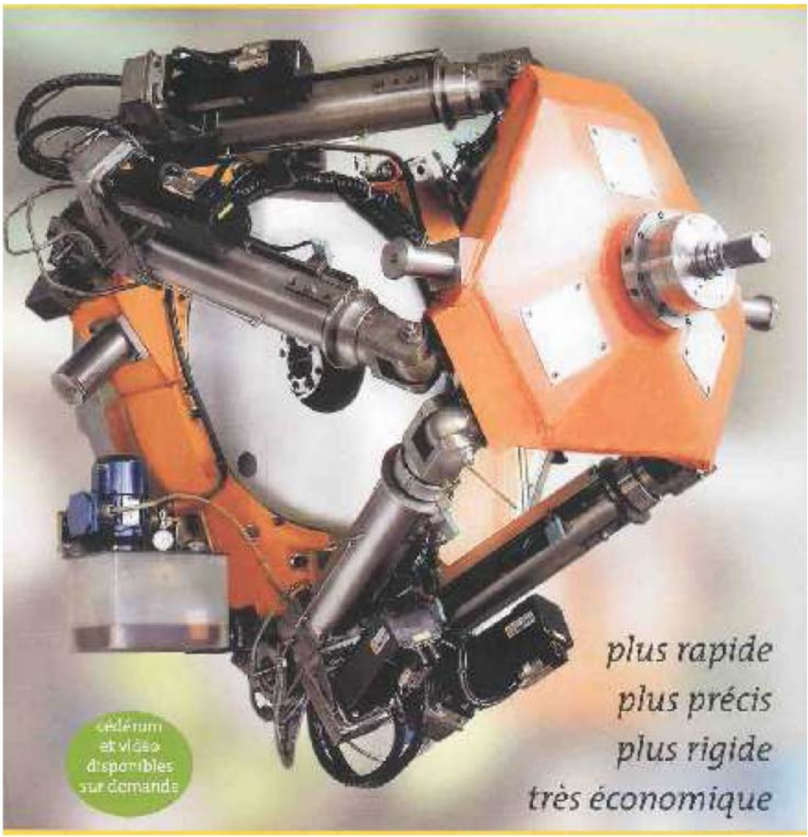
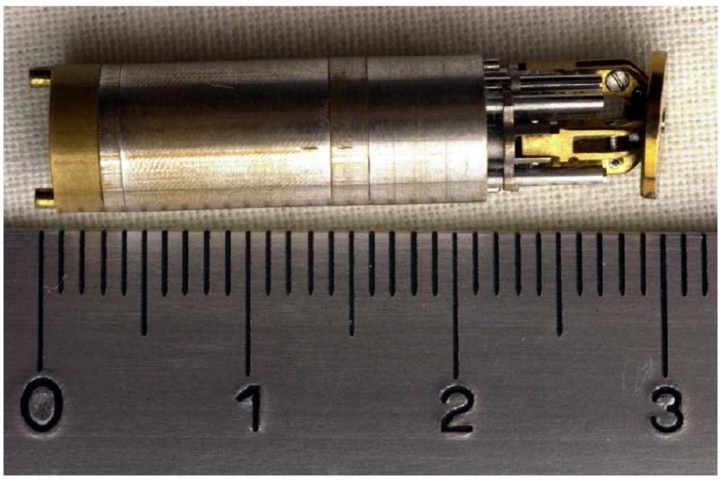


Figure 8 A surgical micro-robot (see online version for colours)



7 Conclusions

Robotics is typical as a field where uncertainties are unavoidable while the safety and reliability of the robot is crucial for many applications. Using statistical approaches to deal with these uncertainties is usually not possible because the safety should be ensured even in the worst case and because many of the uncertainties do not follow classical statistical distributions. We have shown that interval analysis is a tool that allows to deal with these uncertainties. But the efficient use of this tool requires a high level of expertise and a careful analysis of the problem at hand.

Acknowledgements

The work reported in this paper has been partially supported by the European project ARES and by the ANR Siropa project. The author wishes to thank the anonymous reviewers for their useful comments.

References

Angeles, J. (1997) *Fundamentals of robotic mechanical Systems*, Springer, New-York.

Ashokaraj, I., Tsourdos, A., Silson, P. and White, B. (2004, July, 20–23) 'Sensor based robot localisation and navigation: Using interval analysis and extended Kalman filter', *5th Asian Control Conference*, Melbourne.

Carreras, C. and Walker, I.D. (2001) 'Interval methods for fault-tree analysis in robotics', *Transaction on Reliability*, Vol. 50, No. 1, pp.3–11.

Das, I. and Dennis, J.E. (1997) 'A closer look at drawbacks of minimising weighted sums of objectives for pareto set generation in multicriteria optimisation problem', *Structural Optimisation*, Vol. 14, pp.63–69.

Dietmaier, P. (1998, June 29–July 4) *The Stewart-Gough Platform of General Geometry can have 40 Real Postures*, ARK, Strobl, pp.7–16.

Erdman, A.G. (1993) *Modern Kinematics*, Wiley, New-York.

Fang, H. and Merlet, J-P. (2005, August, 2–6) 'Dynamic interference avoidance of 2-dof robot arms using interval analysis', *International Conference on Intelligent Robots and Systems (IROS)*, Alberta.

Fang, H. and Merlet, J-P. (2005, February) 'Multi-criteria optimal design of parallel manipulators based on interval analysis', *Mechanism and Machine Theory*, Vol. 40, No. 2, pp.151–171.

Hansen, E. (1992) *Global optimisation using interval analysis*, Marcel Dekker.

Husty, M.L. (1996, May) 'An algorithm for solving the direct kinematics of Stewart-Gough-type platforms', *Mechanism and Machine Theory*, Vol. 31, No. 4, pp.365–380.

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

Kieffer, M., Jaulin, L., Walter, E. and Meizel, D. (2000, August) 'Robust autonomous robot localisation using interval analysis', *Reliable Computing*, Vol. 6, No. 3, pp.337–362.

Kreinovich, V. (2000) *Optimal Finite Characterisation Of Linear Problems With Inexact Data*, CS-00-37, University of Texas at El Paso, Research Report.

Kreinovich, V., Lakeyev, A., Rohn J. and Kahl, P. (1998) *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer.

Lee, E., Mavroidis, C. and Merlet, J-P. (2004, September) 'Five precision points synthesis of spatial RRR manipulators using interval analysis', *ASME Journal of Mechanical Design*, Vol. 126, No. 5, pp.842–849.

Merlet, J-P. (2001) 'A parser for the interval evaluation of analytical functions and its applications to engineering problems', *Journal Symbolic Computation*, Vol. 31, No. 4, pp.475–486.

Merlet, J-P. (2004) 'Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis', *International Journal of Robotics Research*, Vol. 23, No. 3, pp.221–236.

Merlet, J-P. (2007, June) 'A formal-numerical approach for robust in-workspace singularity detection', *Transaction on Robotics*, Vol. 23, No. 3, pp.393–402.

Merlet J-P., Perng M-W. and Daney, D. (2000, June, 25–29) 'Optimal trajectory planning of a 5-axis machine tool based on a 6-axis parallel manipulator', *ARK*, Piran, pp.315–322.

Moore, R.E. (1979) 'Methods and applications of interval analysis', *SIAM Studies in Applied Mathematics*.

Mourrain, B. (1993, July) 'The 40 generic positions of a parallel robot', in Bronstein, M. (Ed.): *ISSAC'93,* ACM press, Kiev, Ukraine, pp.173–182.

Neumaier, A. (1990) *Interval Methods for Systems of Equations*, Cambridge University Press.

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

Popova, E.D. (2004, April 1) 'Strong regularity of parametric interval matrices', *Thirty Third Spring Conference of the Union of Bulgarian Mathematician*, Borovets.

Raghavan, M. and Roth, B. (1990) 'Kinematic analysis of the 6R manipulator of general geometry', *5th International Symposium of Robotics Research*, Tokyo, pp.263–270.

Rao, R., Asaithambi, A. and Agrawal, S.K. (1998) 'Inverse kinematics solution of robot manipulators using interval analysis', *Journal of Mechanical Design*, Vol. 120, No. 1, pp.147–150.

Rex, G. and Rohn, J. (1998) 'Sufficient conditions for regularity and singularity of interval matrices', *SIAM Journal on Matrix Analysis and Applications*, Vol. 20, No. 2, pp.437–445.

Rohn, J. (2002, September) *Systems Of Interval Linear Equations And Inequalities (Rectangular Case)*, Institute of Computer Science, Academy of Sciences of the Czech Republic, Research Report 875.

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

Tagawa, K. and others. (2001) 'Optimal configuration problem of redundant arms considering endpoint compliance and its solution using interval analysis', *Transactions of the Society of Instrument and Control Engineers*, Vol. 37, p.10.

Tapia, R.A. (1971) 'The Kantorovitch theorem for Newton's method', *American Mathematic Monthly*, Vol. 78, No. 1, pp.389–392.

Wampler, C. and Morgan, A. (1991) 'Solving the 6R inverse position problem using a generic-case solution methodology', *Mechanism and Machine Theory*, Vol. 26, No. 1, pp.91–106.

Wampler, C.W. (1996, April) 'Forward displacement analysis of general six-in-parallel SPS (Stewart) platform manipulators using soma coordinates', *Mechanism and Machine Theory*, Vol. 31, No. 3, pp.331–337.

Wu, W. and Rao, S.S. (2004) 'Interval approach for the modelling of tolerances and clearances in mechanism analysis', *Journal of Mechanical Design*, Vol. 126, No. 4, pp.581–592.

Note

1 www.inria.fr/coprin/logiciels/ALIAS/ALIAS.html