A general methodology for certified evaluation of the performances of parallel robots

J-P. Merlet
INRIA Sophia-Antipolis
France

1 Introduction

Parallel robots offer potentially very attractive performances (such as high accuracy, rigidity and load carrying abilities) and have hence attracted the interest of researchers since approximatively 15 years and of end-users more recently. But at the same time the level of performances of parallel robots is highly sensitive to their dimensioning and high level of performances may be reached only with a careful design of the robot: appropriate design of the basic mechanical components, although necessary, will not be sufficient to obtain good performances as shown by the failure of some recently developed prototypes. Therefore optimal design is a key issue in the field of parallel robots. Unfortunately there is no known method of optimal design that is appropriate for dealing with such complex closed-loop mechanism as parallel robots. Indeed in this type of mechanism there are a large number of parameters, most performances are drastically affected by the pose of the platform and there are large cross-coupling effects between the parameters.

Hence developing an optimal design methodology is a key issue in this field. The most well known design methodology is the cost-function approach [2]. To each design requirement j is associated a numerical index I_j that is minimal for the best robot. The cost function $\mathcal{C}$ is defined as:

$$\mathcal{C} = \sum w_j I_j,$$

where the w_j are weight associated to the I_j. In some sense, the cost function is an indicator of the global behavior of the mechanism with respect to the requirements. As $\mathcal{C}$ is clearly a function of the set of design parameters $\mathcal{P}$, a numerical procedure is used to find the value of the design parameters that minimize $\mathcal{C}$. This approach has several drawbacks:

- the result is heavily dependent upon the weights that are used in the cost-function, and there is no automatic way to find the right weights,
- defining the index I is not always an easy task, for example if we have constraints on the shape of the workspace. Furthermore, some of these index are even very difficult to estimate; for example, some authors have

mentioned the use of a *global conditioning index (CGI)* defined as the average value of the condition number over the workspace of the robot. The condition number itself is obtained as the ratio of the smallest root of a n dimensional polynomial (where n is the number of d.o.f. of the robot) over the largest root of this polynomial. Hence in general the condition number have not an analytical form and consequently computing exactly the CGI is a very difficult task.

- introducing strict requirements in the minimization is difficult, and in any case computer intensive,
- as for any optimization problem, it is difficult to guarantee that the global extremum has been found.
- some of the requirements are antagonistic; for example, it is well known that dexterity is antagonistic with the workspace volume [6]; using both criterion in a weighted sum does not have any physical meaning

But the main difficulty is that the computation of the index for a given geometry must be very efficient as the minimization procedure will use these calculations extensively. Unfortunately, verifying that a given PKS satisfies a single requirement is usually a very complex task and this is this issue that we want to address in this paper.

There are very few works related to performance evaluation in the literature. Without being exhaustive we may mention: stiffness analysis [1], manipulability [4, 13], velocity [5], accuracy [12] and even among this works the proposed method does not always provide exact and guaranteed result.

2 Analysis of the performance evaluation problem

As seen previously any design methodology will make an intensive use of a module that computes the performances of a parallel robot of a given geometry. The performance index are numerous and we may mention, without being exhaustive:

- *kinematics*: workspace, accuracy, maximal motion of the passive joints, dexterity,
- *statics*: load on the platform, stiffness of the robot,

- *dynamics*: maximal velocity and acceleration of the actuator and of the platform, inertia and center of mass,
- *geometrical*: overall size of the robot, of the mechanical components, lack of singularity in a given workspace,
- *technological*: overall information on the actuator, on the sensors and on the passive joints. Indeed the context of the application may impose the use of restricted classes of such components.

We will now consider a specific performance index that will allow us to illustrate the complexity of the task.

2.1 Accuracy analysis

We will consider as an example the evaluation of the accuracy $\Delta\mathbf{X}$ of the positioning of the robot with respect to the accuracy $\Delta\rho$ of the sensors that are used to control the actuated joints. it is well known that these quantities are linearly related by:

$$\Delta\mathbf{X} = J(\mathbf{X})\Delta\rho \tag{1}$$

where $\mathbf{X}$ is the pose of the platform and J is the Jacobian matrix of the robot whose value vary with respect to the pose $\mathbf{X}$. A first difficulty with this performance index is that J is usually a very complex matrix. In most cases only the analytical form of its inverse J^{-1} will be known and a symbolic inversion of this inverse is a computer intensive task that will anyway give an almost useless formula with thousands of terms.

A second difficulty with this index is a general problem: what do we intend to use this index for ? The answers is multi-form:

1. find the worst value of the positioning accuracy *over a given workspace for the robot* or *over all its workspace (i.e. all the poses that can be reached by the platform)*
2. find the average value of the positioning accuracy *over a given workspace for the robot* or *over all its workspace*
3. eventually find the best value of the positioning accuracy over the workspace

All these three items are essential for estimating the real performances of a given robot or to compare the performances between two different geometries. This example allows also to emphasize a major difficulty of the cost function approach: the index regarding accuracy should represent neither only the worst case nor only the average value but a balance between these two values.

2.2 The mathematical problem

We will restrict ourself first to finding the worst and best case positioning accuracy and we will assume that the sensor error $\Delta\rho$ is the interval $[-\epsilon, \epsilon]$. We have to solve the following mathematical problem: find the extremum of each component $\Delta\mathbf{X_i}$ of $\Delta\mathbf{X}$ under the following constraints

1. $|\Delta\rho_i| = \epsilon$ (the maximal error will be obtained when all the sensor errors are maximum)
2. it exists a $\Delta\mathbf{X}$ whose i-th component is the maximum of $\Delta\mathbf{X_i}$ such that $\Delta\rho = J^{-1}\Delta\mathbf{X}$
3. for all the $\mathbf{X}$ satisfying a set of inequalities constraints $W(\mathbf{X}) \leq 0$ (workspace constraints)

Hence the problem to solve is a difficult constrained optimization problem. Note however that if we are able to develop a solver for the above problem, then we are able to use it for computing other performance index. For example the solver may also be used to determine what are the extremum of the joint forces/torques τ for a given load on the platform $\mathcal{F}$ as we have $\tau = J(\mathbf{X})^T\mathcal{F}$.

2.3 The computation accuracy problem

An important point is to analyze what must be the accuracy with which we must solve the problem: shall we always determine exactly the result, at least as far as the use of computer is concerned? Note that exactly means finding the *global* extremum of our optimization with *guarantees* on the result (and this rules out many numerical optimization procedures that may find only a local extremum together with the usual method of sampling the workspace). Clearly the price of computing exactly the result is a large computation time, that must be avoided if possible.

Intuitively we can see that for some cases it is not always necessary to determine exactly the result. For example assume that the purpose of calculating the worst case accuracy is to determine what must be the sensor accuracy β for reaching a defined accuracy γ on the positioning of the platform in order to be able to choose the sensor among a set of n available sensors with errors $\epsilon_1, \ldots, \epsilon_n$ (ordered by increasing value). We will compute first the worst case accuracy γ_1 for a unit value of the sensor error ϵ, from which we will deduce that the maximal sensor error β for reaching the positioning accuracy γ will be γ/γ_1. But it is not necessary to compute *exactly* the value of γ_1. Indeed if we can guarantee that γ_1 is lower than $\overline{\gamma_1}$ and greater than $\underline{\gamma_1}$ such that we have an ϵ_i with:

$$\gamma/\overline{\gamma_1} \geq \epsilon_{i-1} \quad \gamma/\underline{\gamma_1} \leq \epsilon_i$$

then we have determined that the sensor with accuracy ϵ_i will be the right choice. Note however that for finding the right sensor it is necessary to be able to adjust the maximal value of the difference $\overline{\gamma_1} - \underline{\gamma_1}$.

Hence it may be sufficient to find an interval that contain the exact value of γ_1, as soon as the result is *guaranteed*. In summary:

- the design methodology should take into account the fact that computing exactly the value of most performance index will be computer intensive and should be avoided as much as possible
- the result of the performance evaluation must be guaranteed
- the result of the performance evaluation may be not exact: an interval containing the exact value may be sufficient as soon as:
  - the exact value is guaranteed to lie within the interval
  - the maximal width of the interval may be adjusted
- if possible performance evaluation procedures should be designed in such way that:
  - approximate solution can be found
  - the computation time must be dependent upon the accuracy with which the result is calculated
  - previous runs of the module with a given accuracy on the result may be re-used if the accuracy is decreased to avoid unnecessary computations

3 Genericity for performance evaluation

As we have seen previously there is no difference, in the mathematical sense, between solving the worst case accuracy problem and finding the extremal forces/torques for the actuated joints. In my opinion there are numerous problems in the field of performance evaluation that are similar in term of structure. Hence solving a few key problems that have a standard form (that will be called the *standard verification form* (SVF)) may be sufficient to solve most of the usual requirements that have to be considered when designing a parallel robot as soon as first the requirements may be translated into a SVF and if it exists an appropriate solver (called the *standard verification form solver* (SVFS)).

Another important point in this field is the *genericity* of the approach i.e. a performance evaluation procedure should work for the most usual requirements but also *for any mechanical architecture of parallel robots.* Indeed there is a large architectural diversity between parallel robots (over 100 designs have been proposed in the literature) and it is unrealistic to develop a performance evaluation procedure dedicated to each of them.

When addressing a performance evaluation problem there are three major steps:

1. reducing the problem to a SVF
2. finding the mathematical elements that are used in the SVF, according to the structure of the robot
3. solving the SVF using the SVFS

Hence the topology of the robot is only important at step 2 and it is therefore crucial to clearly separate this step from step 1 and 3. If we consider the accuracy problem this step consists in finding the inverse Jacobian matrix of the robot. Hence a fully generic performance evaluation procedure should be able to provide such element to the SVF. This may eventually be done automatically (probably by using symbolic computation) if *a standard definition format allows one to describe any parallel structure.* Hence the definition of a standard definition language is also a key point for the optimal design of parallel robot. Note also that symbolic computation is a necessary tool that will be used for all three steps

4 Standard verification form solvers

A crucial point in the development of a generic performance evaluation procedure is the module that allow the solving of the SVF. As we have seen previously many such problems may be reduced to a complex constrained optimization problem. In my opinion there is not many hope to find a method for finding the exact solution that may work whatever is the performance and the topology of the robot. Two alternative approaches may be considered: hybrid solvers and interval analysis.

4.1 Hybrid solvers

Hybrid solvers uses a mix of exact and approximate solving: their purpose is to solve the SVF optimization problem with an accuracy ϵ that is fixed in advance. Remember that in general we will have to solve the optimization problem when the pose vector $\mathbf{X}$ is constrained i.e. the pose of the platform belongs to some workspace : for the sake of simplicity we will assume here that the workspace is defined by a set of intervals such that for all components X_i of $\mathbf{X}$ we have $X_i \in [\underline{X_i}, \overline{X_i}]$. If we assume that $\mathbf{X}$ is of dimension n, then it may be possible to find the exact solution to the optimization problem when the first $m < n$ components of $\mathbf{X}$ have a fixed value.

The first step of an hybrid solver consists in solving the SVF optimization problem when the first $m < n$ components of $\mathbf{X}$ are fixed to their lower bound ($X_i = \underline{X_i}$). This gives an initial value S_0 for the SVF optimization problem. Then a second optimization problem is solved: we consider the SVF optimization problem when the pose parameters having a fixed value are $X_1 = \underline{X_1}+\alpha$, $X_i = \underline{X_i}$ for i in [2,m] and we find an α such that the result S_1 is guaranteed to verify $|S_0 - S_1| \leq \epsilon$. As soon as such an α has been determined we solve the SVF optimization problem with the new value of X_1: according to the result of this solving we update if necessary the value of S_0. We then iterate the process until we have $X_1 \geq \overline{X_1}$. At this point we have solved the SVF optimization problem with an error at most ϵ when $m-1$ components of $\mathbf{X}$ have a fixed value, the result being the updated value of S_0.

We then solve a third optimization problem which is to find a β such that the result S_2 of the SVF optimization problem for the pose parameters being $X_i = [\underline{X_i}, \overline{X_i}]$ for $i = 1, 3 \ldots n$ and $X_2 = \underline{X_2} + \beta$ verifies $|S_2 - S_0| \leq \epsilon$. As soon as the value of β has been found we repeat the first step with as value for $X_2 = \underline{X_2} + \beta$. The whole process is then repeated until the full workspace has been explored.

Such an hybrid solver has been used to determine the worst case accuracy and maximal joint forces for a Gough-Stewart platform [8, 9] when the orientation of the platform was supposed to be constant. If the location of the center of the platform is defined by the variables x, y, z it was possible to show that the SVF can be solved exactly for a fixed value of y, z, x being constrained to lie in a range $[\underline{x}, \overline{x}]$. Using the technique described above we are able to solve the SVF problem when the center of the platform is restricted to lie within a square defined by $x \in [\underline{x}, \overline{x}]$, $y \in [\underline{y}, \overline{y}]$, $z = \underline{z}$. By solving the third optimization problem we determine a new fixed value for z, then we solve the SVF problem for the square at the new altitude. This process is repeated until the whole workspace has been explored.

4.2 Interval analysis

Interval analysis is a powerful method initially proposed by Moore [11], whose application to global optimization problems has been emphasized by Hansen [3]. Let us illustrate this method on a simple example: let f be the function x^2-2x and assume that we are looking for the maximal value of f when x is in the range $[3, 4]$.

Intuitively it is easy to see that if x is in [3,4], then x^2 is in [9,16] and similarly $-2x$ is in the range [-8,-6]. Now consider the sum of 2 intervals $A = [\underline{a}, \overline{a}]$, $B = [\underline{b}, \overline{b}]$. It may be seen that $A + B = [\underline{a} + \underline{b}, \overline{a} + \overline{b}] = C$, which means that for any value of x in A and y in B, then $x + y$ lie in C. In our case we will write $f([3, 4]) = [9, 16] + [-8, -6] = [1, 10]$. The resulting interval defines therefore lower and upper bound for the values of f: we may guarantee that for any x is [3,4], $1 \leq f(x) \leq 10$ and hence that the maximum of f is at least 1. Now assume that we want to determine the maximal value of f with an accuracy 0.1. The result obtained up to now does not allow to get such accuracy as the difference between the lower and upper bound is larger than this value. We will now bisect the initial interval in two new intervals $[3, 3.5]$, $[3.5, 4]$ and repeat the interval evaluation for each of them. We found $f([3, 3.5]) = [2, 6.25]$ and $f([3.5, 4]) = [4.25, 9]$. Hence the maximal value of f is at least 4.25. We then repeat the bisection process on each interval $[\underline{x}, \overline{x}]$ for x such that the upper bound of $f([\underline{x}, \overline{x}])$ is greater than the current maximum value+0.1 (for example the interval [3,3.1] will not be bisected as $f([3, 3.1]) = [2.8, 3.61]$ and $3.61 < 4.25 + 0.1$) and the algorithm will stop when all the intervals have been processed. Note that the above description illustrate only the basic principle of interval analysis and that, although this basic method will work, it may be deeply improved.

Note also that a similar approach may be used to solve $f(x) = 0$: in that case a solution will be obtained if the width of the interval for x is lower than a given threshold and the interval evaluation of f contains 0 (in our example as $f([3, 4]) = [1, 10]$ does not contain 0 we may state that there is no solution of $f(x) = 0$ for x in the range [3,4]).

The advantages of interval analysis are:

- this method works for all the classical mathematical functions such as $\sin, \cos, \sinh, \ldots$
- it may be implemented to take into account numerical round-off errors and is therefore safe from a numerical view point
- it allows to determine *global* extrema

- it may be implemented in a distributed way: assuming that n computers are available each of them may process a set of intervals and return the result to a master program, thereby drastically increasing the efficiency of the method.

In my opinion interval analysis is a very promising method to implement SVFS: we have started implemented in our `ALIAS` C++/Maple library [1] a set of such methods and this library has been used to solve various problems related to parallel robots:

- singularity detection [10]: we have implemented a generic procedure to detect singularity within the workspace of any 6 dof parallel robot. Symbolic computation is used to calculate the determinant of the matrix J^{-1} and a C++ program allow to determine if this determinant may be 0 within some given workspace. The workspace may be defined either as a set of intervals for the pose parameters $\mathbf{X}$ or as any pose that satisfy a set of inequalities constraints $G(\mathbf{X}) \leq 0$ (for example all the poses such that the joint coordinates satisfy some inequalities)
- trajectory verification [7]: interval analysis is used here to determine if an arbitrary trajectory fully lie within the workspace of a parallel robot i.e. the joint coordinates satisfy a set of constraints. The trajectory is defined as a set of time-dependent analytical functions, one for each of the pose parameters.

5 Conclusion

Performance evaluation is a key issue for the development of efficient parallel robots. A bad point is that this is also a very complex issue that has been carefully analyzed in this paper. In my opinion the main conclusions are:

- the design methodology should take into account the fact that computing exactly the value of most performance index will be computer intensive and should be avoided as much as possible
- the result of the performance evaluation must be guaranteed
- the result of the performance evaluation may be not exact: an interval containing the exact value may be sufficient

[1] http://www-sop.inria.fr/coprin/logiciels/ALIAS/ALIAS.html

We have then proposed a framework to establish a performance evaluation procedure that is generic (i.e. will work for most usual design requirements and for any type of parallel robot). Implementing such procedure will require a huge effort: hence this will be possible only through a collaborative work between academia, companies and end-users.

References

[1] El-Khaswneh B. and Ferreira P.M. Computation of stiffness and stiffness bounds for parallel manipulators. *Int. J. of Machine Tools & Manufacture*, 39(2):321–342, February 1989.

[2] Erdman A.G. *Modern Kinematics.* Wiley, New-York, 1993.

[3] Hansen E. *Global optimization using interval analysis.* Marcel Dekker, 1992.

[4] Hong K.S. and Kim J-G. Manipulability analysis of a parallel machine tool: application to optimal link length design. *J. of Robotic Systems*, 17(8):403–415, 2000.

[5] Huynh P. and Arai T. Maximum velocity analysis of parallel manipulators. In *IEEE Int. Conf. on Robotics and Automation*, pages 3268–3273, Albuquerque, April, 21-28, 1997.

[6] Ma O. and Angeles J. Optimum architecture design of platform manipulator. In *ICAR*, pages 1131–1135, Pise, June, 19-22, 1991.

[7] Merlet J-P. A parser for the interval evaluation of analytical functions and its applications to engineering problems. *J. Symbolic Computation*, 31:475–486, 2001.

[8] Merlet J-P. Efficient computation of the extremum of the articular velocities of a parallel manipulator in a translation workspace. In *IEEE Int. Conf. on Robotics and Automation*, pages 1976–1981, Louvain, May, 18-20, 1998.

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

[10] Merlet J-P. and Daney D. A formal-numerical approach to determine the presence of singularity within the workspace of a parallel robot. In F.C. Park C.C. Iurascu, editor, *Computational Kinematics*, pages 167–176. Seoul, May, 20-22, 2001.

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

[12] Ropponen T. and Arai T. Accuracy analysis of a modified Stewart platform manipulator. In *IEEE Int. Conf. on Robotics and Automation*, pages 521–525, Nagoya, May, 25-27, 1995.

[13] Zhang Y., Duffy J., and Crane C. The optimum quality index for a spatial redundant 4-8 in parallel manipulator. In *ARK*, pages 239–248, Piran, June, 25-29, 2000.