Proceedings of the 11th World Congress in Mechanism and Machine Science
August 18-21, 2003, Tianjin, China
China Machinery Press, edited by Tian Huang

Getting exact information from the inverse jacobian matrix of parallel and serial robots

J-P. Merlet
INRIA, 06902 Sophia Antipolis Cedex, France
e-mail: Jean-Pierre.Merlet@sophia.inria.fr

Abstract:
The inverse jacobian matrix of a robot contains highly useful information about the performances of the robot, for example about its accuracy and dexterity. Unfortunately these information are available only mostly through the eigenvalues of this matrix which are pose and design parameters dependent. We present interval analysis based algorithms that allow to extract useful information from any matrix such as minimal and maximal eigenvalues and minimal condition number.

Keywords: conditioning, isotropy, jacobian matrix

1 Introduction

The jacobian matrix J of a robot relates linearly the joint velocities $\dot{\Theta}$ and the cartesian velocities $\dot{X}$:

$$\dot{X} = J(X)\dot{\Theta} \tag{1}$$

A first order approximation allows also to relate the errors ΔX on the positioning of the end-effector to the error $\Delta\Theta$ in the joint variable measurements:

$$\Delta X = J(X)\Delta\Theta \tag{2}$$

Let us introduce the joint errors space as a n dimensional space (if the robot has n joints) where each dimension corresponds to one of the joint errors. In the joint errors space a point correspond to a set of values for the joint errors. If we assume now that the amplitude of all the joint errors is bounded by an arbitrary value (e.g. 1) we have $||\Delta\Theta||^2 \leq 1$ and all the possible joint errors are included in a hyper-cube. We get:

$$||\Delta\Theta||^2 = ||\Delta X||J^{-1}J^{-T}||\Delta X|| \leq 1 \tag{3}$$

Equation (3) indicate that the joint errors space hyper-cube is mapped into a convex hyper-polyhedra in the cartesian error space. The largest (smallest) eigenvalue of $M = J^{-1}J^{-T}$ will provide the largest (lowest) dimension of the hyper-polyhedra i.e. a value closely related to the largest (smallest) error for the position of the end-effector, while the condition number of M (i.e. the ratio of the smallest eigenvalue over the largest one) will reflect the shape of the polyhedra: if 0 the polyhedra is reduced to a line and the robot is in a singular position and if 1 the polyhedra is an hyper-cube. Hence interesting information are provided by the eigenvalues of M. Many authors have consequently suggested to use this matrix for establishing a quality index for a robot: either through the determinant of this matrix [6] or its condition number [3]. But there are major drawbacks with M:

- this matrix depends upon the pose and the geometry of the robot.
- the eigenvalues that are used to calculate the index may be obtained numerically is the matrix is numerical. Otherwise a closed-form expression of the eigenvalues may be obtained only by studying the characteristic polynomial of M which has the same degree than the dimension of the matrix. Hence for robot having more than 4 d.o.f. it will not be possible to determine an analytical form for the eigenvalues. For robot having up to 4 d.o.f. although the roots may be known analytically the calculation o the index may not be trivial

In general it will be possible to compute numerically the index for a given robot at a given pose. But the indexes have for purpose to give information on the behavior of the robot not only in a single pose but over its whole workspace. Computing these indexes in this view is equivalent to solve a difficult constrained optimization problem. A critical point is that the result must be guaranteed: the algorithm should provide a *global* minimum or maximum and not a local one. The purpose of this paper is to explain how to compute numerically the value of various indexes and even how to deal with design problem when constraints are given on the indexes. All these algorithms are based on an interval analysis approach.

2 Interval analysis

Interval analysis is a powerful method initially proposed by Moore [2] in which real numbers are substituted by intervals. 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 solutions of $f = 0$ 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]: this means that if x has a particular value in the range [3,4], then $f(x)$ has a value in the range [9,16] (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 lower and upper bounds for the values of f: we may guarantee that for any x in [3,4] we have $1 \leq f(x) \leq 10$. As 0 is not included in the final interval we may state that there is no zero of f for x in the range [3,4]. Note that the bounds provided by interval arithmetics are overestimated, the true range of $f(x)$ being [3,8] (but the amplitude of the overestimation will decrease with the width of the interval for x). However, this does not affect the validity of the conclusion.

This method works for all the classical mathematical functions such as $\sin, \cos, \sinh, \ldots$. Furthermore this method may be implemented to take into account numerical round-off errors and is therefore safe from a numerical view point. Hence interval analysis is an elegant method to determine in a simple way lower and upper bounds of any arbitrary expression. In this paper an interval will be denoted by upper-case letters and the lower bound and upper bound of the interval X will be denoted $\underline{X}, \overline{X}$.

3 Minimal and maximal eigenvalues

Let us assume that M is function of a set of n parameters $p_1, \ldots, p_n$, all supposed to lie in a set S_0 of given ranges $P_1^0, \ldots, P_n^0$ (hence M represents a set of matrices). These parameters may be either the pose of the robot, a set of design parameters or a mix of both. Our aim is to compute the minimal eigenvalues e_m of M up to a pre-defined accuracy ϵ. In other words we will determine a number e'_m, which will be the lowest eigenvalue of one particular matrix in M, such that $|e'_m - e_m| \leq \epsilon$. In term of optimization if P denotes the characteristic polynomial of M we want to determine a set (S, x) such that x is minimized subject to the constraints $P_S(x) = 0$ and $S \in S_0$. It may be seen that this is not a trivial constrained optimization problem.

It may be thought that for robot with up to 4 d.o.f. this problem may be solved as analytical forms of the eigenvalues are known. This is not a very good idea:

- the analytical form of the root is usually very badly conditioned
- its is difficult to use the analytical form for determining the minimal eigenvalue

We define now the set of unknowns $U = \{y, p_1, \ldots, p_n\}$ where the first unknown represent a potential root for P. A *box* will be defined as a set of intervals $\{Y, P_1, \ldots, P_n\}$, each element of this box being associated to the corresponding element in U.

Consider now the characteristic polynomial P of M: the coefficients of P can be calculated as a function of the n parameters. Using interval analysis we are able to compute the bounds of all these coefficients. Then by using classical results of algebraic geometry (such as Laguerre or Cauchy theorems) and classical results for the bounds of eigenvalues (Gerschgorin circles [1], Kharitonov polynomials [4], ...) we are able to determine lower and upper bounds for the eigenvalues of M. Hence we get an interval $Y_0 = [a, b]$ such that all the roots of any P_S where $S \in S_0$ are included in $[a, b]$. More generally for a box B_j we will denote by $G(B_j)$ the interval that include all the roots of the polynomials $P(B_j)$ whatever are the values of the parameters in B_j.

We will denote by $\mathcal{P}(B)$ the interval evaluation of the polynomial P for the box B: if this evaluation does not include 0, then there is no value of x in Y such that $P(x) = 0$, whatever are the parameters values in $P_1, \ldots, P_n$. During the calculation we will maintain an estimate e'_m of e_m: this estimate will be initialized by taking random values for the parameters within their range, computing numerically the minimal root of P and assigning this value to e'_m . A list $\mathcal{L}$ of box will be used during the calculation. This list is initialized with the box $\{Y_0, P_1^0, \ldots, P_n^0\}$. The total number of box in $\mathcal{L}$ will be denoted k while B_j will represent the j-th box $\{Y_j, P_1^j, \ldots, P_n^j\}$ in the list. During the calculation we will process each box in $\mathcal{L}$ and the number of the current box being processed will be denoted by i. The basic algorithm proceeds along the following steps:

1. if $i > k$, return e'_m
2. if $\underline{G(B_i)} \geq e'_m - \epsilon$, then $i = i+1$ and go to step 1
3. if $\underline{Y_i} \geq e'_m - \epsilon$, then $i = i+1$ and go to step 1
4. if $\overline{Y_i} \geq e'_m - \epsilon$, then change the upper bound of Y_i to $e'_m - \epsilon$
5. if $\mathcal{P}(B_i)$ does not include 0, then $i = i+1$ and go to step 1
6. select a few random points in B_i and compute numerically the minimal roots of P at these points. If one of these roots is lower than e'_m, then update e'_m
7. choose one variable l in the box and bisect its range. This create 2 new ranges V_1, V_2 for the variable l and we create 2 new boxes from the original B_i: they have the same range for all the variables, except for the variable l, the first box having range V_1, while the other has range V_2. B_i is replaced by the first new box, while the second new box is added at the end of the list. k is updated to k+1 and we go to step 2.

This basic algorithm may be easily explained:

- step 1 is the completion test of the algorithm: all the boxes have been processed and we may guarantee that e'_m is a good approximation of e_m.
- steps 2, 3, 5 are test that allow to reject boxes that cannot contain polynomials whose lowest root will be lower than $e'_m - \epsilon$
- step 4 is an update test: the first variable is a box is the potential minimal root of P and we are interested only in root improving the current minimum

- step 6 is an update procedure for the minimum. The current box defines a set of polynomials: we just consider a few polynomials in this set and update the current minimum if necessary

- step 7 is the reduction step. We have not been able to determine if in the set of polynomials defined by B_i there may be a polynomial having a root lower than the current minimum. So we create 2 new boxes from the initial one, each of these boxes having a smaller width than the original one

Even this basic algorithm has advantages. First we are not limited in term of complexity of the characteristic polynomial: the only requirement is that we must use a numerical procedure that can compute safely the roots of a numerical polynomial or the eigenvalue of a matrix (step 6). Note also that, like most algorithms based on interval analysis, the structure of the algorithm is appropriate for a distributed implementation: each box may be processed by a slave computers for a few iterations, the result being returned to a master program that manages the list $\mathcal{L}$, the result and the load of the slave computers. Note that small changes in the algorithm allows to compute as well the maximal root or both the minimal and maximal roots. Finally the algorithm may be extended to deal with additional constraints on the parameters. For example assume that we are looking for the minimal root of the polynomial when the workspace is a sphere (supposed to be centered at the origin and to be of radius R). Here we have two possibilities:

- either we choose as pose parameters the distance to the center of the sphere and the latitude and longitude angles

- or we use the cartesian coordinates for the end-effector with the constraints $F = x^2+y^2+z^2 \leq R^2$

In the later case we first add a step after step 1 a further test on the interval evaluation of F: if $\underline{F}$ is greater than R^2, then the box is rejected. Then we add a step before step 6: if $\overline{F}$ is greater than R^2 we skip step 6 and go to step 7.

For researchers familiar with branch-and-bound algorithm this basic algorithm may look trivial. But drastic differences in efficiency may be observed between 2 algorithms using the same principle. The important points are the computation of $G(B_i)$, the "reject" test of steps 2,3,5 and the choice of the bisected variable in step 7.

Finally our algorithm allows to deal with the case where the matrix M is highly complex, thereby prohibiting to compute the analytical form of the characteristic polynomial. Our implementation allows to deal with that case as soon as we are able to compute an interval evaluation of the coefficients of M.

4 Maximal region for given bounds on the roots

In this section we will present an algorithm that allows to determine an approximation of all the parameters values such that the corresponding characteristic polynomials have all its real roots within a given range $[\alpha, \beta]$. The result of this algorithm will be a list of boxes, called the *solution list*.

The principle of this algorithm is the same than for the algorithm presented in the previous section except that the variable in the box are now only the n parameters. Step 1 is identical while the test in step 2 will be: if $\underline{G(B_i)} \geq \beta$ or $\overline{G(B_i)} \leq \alpha$ which will allow to eliminate boxes with non valid roots. Steps 3 and 4 will be substituted by a new test based on Budan-Fourier algorithm. This algorithm allows to count the number of roots of a polynomial in a given range [a,b] using only the polynomial evaluation at a and b and the evaluation of the derivatives of this polynomial at the same points. The number returned by this algorithm is the number of real roots in [a,b] up to a power of 2. Hence if this number is even we can guarantee that there is at least one root in [a,b]. We design a test $BF(B_i)$ that use the Budan-Fourier method for the interval $[\infty, \alpha]$ and $[\beta, \infty]$: this test returns -1 if Budan-Fourier has allowed to determine that there is at least one root in one of these ranges. Alternatively this test returns 1 if Budan-Fourier has allowed to determine that there is no root of the polynomial in both ranges. Step 3 is now:
if $BF(B_i)$= -1 then $i = i+1$ and go to step 1 if $BF(B_i)$= 1 then include B_i in the solution list

Step 5 is also modified: the interval evaluation of the polynomial is computed for the variable being in the range $[\alpha, \beta]$. Step 6 is changed to a test that will allow to control the quality of the approximation: if the width of B_i is lower than a given threshold ϵ, then the box is discarded.

The solution list is an approximation (as we have neglected the boxes with width lower than ϵ) of all the possible parameters values such that all the real roots of the characteristic polynomial of M lie in the range $[\alpha, \beta]$. It may be used, for example, to determine in which part of the workspace of the robot the dexterity index is acceptable (see the example in the last section).

It is possible to determine the quality of the approximation by calculating the total volume V_{in} of the box in the solution list and the total volume V_{out} of the boxes that have been neglected during the calculation. The ratio V_{out}/V_{in} is a good index for measuring the quality of the approximation. Note that the algorithm allows to incrementally improve the quality of the approximation. Indeed assume that during a first run the neglected boxes are stored in a file: if the approximation index is not satisfactory we will perform a second run of the algorithm with a lower value for ϵ. But instead of using the same initial box than during the first run we will use the list of rejected boxes, thereby avoiding to repeat the calculation of the solution boxes that have already been obtained during the first run.

5 Minimal condition number

The purpose of this section is to present an algorithm for determining the minimal value of the condition number of M, which is here defined as the ratio between the minimal and maximal eigenvalues. For simplicity we will assume here that all the eigenvalues are positive. The structure of the algorithm is basically similar to the one presented in the previous sections: we outline here only the main differences.

A box in this algorithm is defined as $U = \{y, p_1, \ldots, p_n\}$ where y represent the largest eigenvalue, e_m is supposed to be the minimal condition number and e'_m the current estimation of e_m. The reject test ensure first that the range for y may contain a root of P by using the interval evaluation of P and eliminates boxes that do no fulfill this condition. Then the reject test uses the Budan-Fourier method to determine to determine the number of roots of P in the range $[\overline{y}, \infty[$: if this number is larger than 0 the box is eliminated as the range for y does not define a range for the maximal eigenvalue. In a third step the reject test determine if the polynomials in P may have a root lower than $\underline{y}e'_m$ by using both the interval evaluation of P and Budan-Fourier for the range $]-\infty, \underline{y}e'_m]$: if this test is negative the box is eliminated. For each box which is not eliminated by the reject test we update the value of e'_m by computing the condition number for specific polynomials in P obtained by taking for the parameters some values that lie within within the range defined by the box.

6 Implementation and experimental tests

All the algorithms presented in the previous sections are available within the `ALIAS` library developed by the COPRIN project of INRIA. `ALIAS` is a C++ library that implements algorithms based on interval analysis for the solving of large categories of problems. One of the feature of `ALIAS` is its Maple interface. For the problem mentioned previously the end-user has only to create a Maple program that compute the matrix M and then call a specific procedure of the `ALIAS` maple interface. This procedure will create automatically the necessary C++ code, compile and execute the program and then returns the result to Maple.

As a test example we have used the 3 d.o.f. parallel robot Orthoglide, figure 1 [5]. The structure of the robot allows to control the translation of its end-effector while its orientation remains constant. As the velocity $\dot{\rho}$ of the linear actuator are bounded we have a constraint on the end-effector velocities $\dot{X}$:

$$||\dot{\rho}|| \leq 1 \Rightarrow \dot{X}^T (JJ^T)\dot{X} \leq 1$$

In the joint velocities space we have hence a cube that is linearly mapped to a convex polyhedra in the end-effector velocities space. The three eigenvalues of the matrix J^TJ are the length of the principal axis of the polyhedra and are a good indication of the velocity

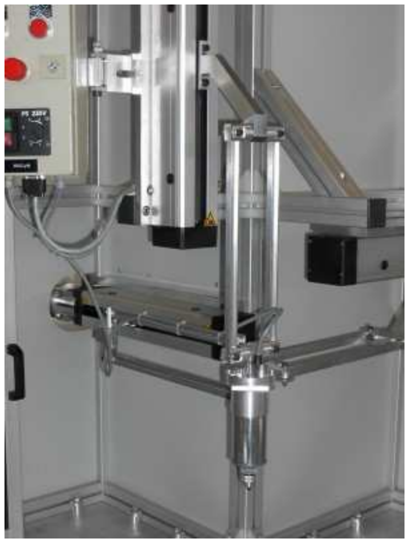

Figure 1: The Orthoglide robot

transmission factor. A low transmission factor indicates that the velocities of the actuators is used mainly to preserve the geometry of the mechanism and not for moving the end-effector. On the other hand a large transmission factor indicates that the motion of the end-effector will be very sensitive to a change in the actuator velocity. Hence we want to determine the region of the 3D workspace, called the *useful workspace*, where the transmission factor will lie within a predefined range (in our case the range is [0.25,4]).

The workspace of this robot is defined as the intersection of three cylinders defined by:

$$x^2 + y^2 \leq 1 \quad x^2 + z^2 \leq 1 \quad y^2 + z^2 \leq 1 \tag{4}$$

Our initial box (i.e. the set of ranges for x, y, z) is defined as $\{[-1, 1], [-1, 1], [-1, 1]\}$ and we have established an approximation of the useful workspace as a list of boxes in the 3D space. For any pose in any box in the list the 3 transmission factors are guaranteed to lie within the range [0.25,4] and the constraints (4) are satisfied.

The computation is directly done within Maple: using the `ALIAS` Maple interface it is possible to define the constraint equations that define the workspace, to compute the characteristic polynomial that is of interest and then to produce automatically the necessary C++ code. A typical Maple code for a characteristic polynomial `P` will be:

```
eq1:=x^2+y^2-1<=0:
eq2:=x^2+z^2-1<=0:
eq3:=z^2+y^2-1<=0: epsilon:=0.1:
`ALIAS/opt_sol_min`:=0.25:
`ALIAS/opt_sol_max`:=4:
MinMax_Polynom_Area([eq1,eq2,eq3,P],
      [lambda,x,y,z],
      [[-10,10],[-1,1],[-1,1],[-1,1]],
      epsilon);
```

Figure 2 shows the part of the useful workspace located between $-0.3 \leq x \leq 0$. We may distinguish 3 components: the neglected boxes that correspond to the border of the workspace, the approximation of the useful workspace (at the center of the figure) which is surrounded by a set of neglected boxes. Figure 3 shows a cross-section of the useful workspace for a fixed value of z. The dark grey boxes are the neglected boxes while the light grey one represent the approximation of the useful workspace.

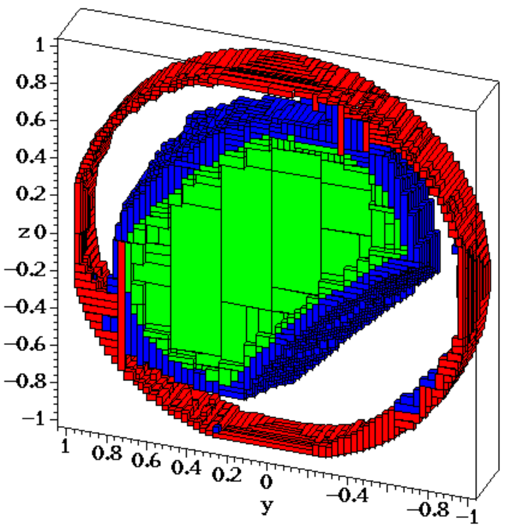

Figure 2: Useful workspace for $-0.3 \leq x \leq 0$.

Table 1 indicates the values of the volume V_u of the useful workspace, the value of the volume V_n of the neglected boxes and the computation time for various values of ϵ. For the computation time the value between parenthesis indicates the time obtained when the incremental mode is used. It must be noted in that case that we should refine the notion of rejected box: indeed such a box will occur either when a box includes part of the border of the useful workspace or when it includes part of the border of the total workspace (i.e. as defined by the constraint equations (4)). In the table the value of V_n indicated between parenthesis indicate the volume of the neglected boxes that do not include elements of the total workspace border.

ϵ	V_u	V_n	Time (s)
0.2	0.85431	3.7722 (2.2968)	1893
0.1	1.16789	2.39753 (1.5951)	10284 (6501)
0.05	1.36109	1.34454 (0.93105)	33328 (27187)

Table 1: Volume of the useful workspace and of the neglected boxes for various values of ϵ

7 Conclusion

Many performance index of a robot rely on the determination of the eigenvalues of a matrix which is a function of the geometry and of the pose of the robot. Although these index can be computed numerically at

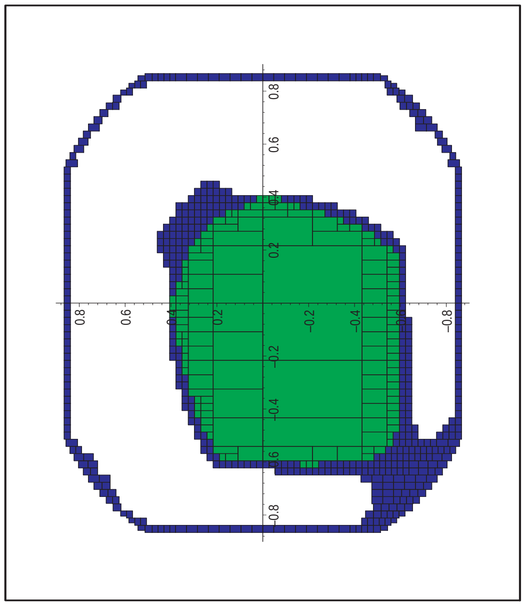

Figure 3: Cross-section of the useful workspace for $z = 0.5$.

a specific pose we are mainly interested in their minimal and maximal values over the workspace of the robot. We have shown that interval analysis allows to calculate these index whatever is the geometry of the robot and that this method allows to consider complex workspace.

Acknowledgments

The author would like to acknowledge the financial support of the CNRS ROBEA project.

References

[1] Golub G.H. and Van Loan C.F. *Matrix computations*. John Hopkins University Press, 1996.

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

[3] Salisbury J.K. and Craig J.J. Articulated hands: force control and kinematic issues. *Int. J. of Robotics Research*, 1(1):4–17, 1982.

[4] V.L. Kharitonov. Asymptotic stability of an equilibrium position of a family of systems of linear differential equations. *Differential'ye Uravneniya*, 14:2086–2088, 1978.

[5] Wenger P. and Chablat D. Kinematic analysis of a new parallel machine-tool: the Orthoglide. In *ARK*, pages 305–314, Piran, June, 25-29, 2000.

[6] Yoshikawa T. Manipulability of robotic mechanisms. *The Int. J. of Robotics Research*, 1(1), 1982.