Implementation with Orccad of a Method for Smooth Singularity Crossing in a 6-DOF Manipulator

Konstantin Kapellos, Bernard Espiau

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Implementation with Orccad of a Method for Smooth Singularity Crossing in a 6-DOF Manipulator

Konstantin Kapellos, Bernard Espiau

N° 2654

Septembre 1995

PROGRAMME 4

ISSN 0249-6399

INRIA
RHÔNE-ALPES

Implementation with Orccad of a Method for Smooth Singularity Crossing in a 6-DOF Manipulator

Konstantin Kapellos, Bernard Espiau

Programme 4 — Robotique, image et vision
Projet Bip

Rapport de recherche n° 2654 — Septembre 1995 — 37 pages

Abstract: This paper addresses the problem of kinematics singularity crossing for robot manipulators. The proposed method is based on the task-function approach ([5]) and is quite general. It has already be applied to the simulated case of a simple planar redundant robot ([6]), and we propose here a complete description of its extension to a six degrees of freedom robot. The paper is organized in two main parts.

After a brief state-of-the-art, we describe in details the general proposed method. From the control point of view, it relies on the framework of the computer torque approach, expressed in a general way. The control objective is defined under the form of an output regulation problem. This goal can be shown in the present case to correspond to the minimization of a quadratic cost function, which blends a trajectory tracking requirement in translation and orientation and a regularizing term. This last term expresses a "safe" behavior of the robot in the neighborhood of a singularity, and depends on a time-varying function which may sometimes be interpreted as an image of the robot joint velocity. The relative weights of the two components of the cost function vary with respect to the proximity of the singularity. What should be emphasized is that this method includes the wellknown damped least-squares approach as a particular case. Another point to mention is that the method provides a general framework to the approach reported in [7], which leads to similar results.

The second part of the paper is devoted to specification and implementation issues. We use for that purpose the ORCCAD environment ([8]) and we show how the method can be practically implemented. We present its functional decomposition, including temporal aspects. We also show what are the associated discrete events to handle, and how they can be controlled by an automatically generated automaton. Finally, results concerning wrist, elbow and shoulder singularity crossing for an AID robot are given.

Key-words: Robotics, Control, Manipulators, Singularities

(Résumé : tsvp)

Unité de recherche INRIA Rhône-Alpes
46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1 (France)
Téléphone : (33) 76 57 47 77 – Télécopie : (33) 76 57 47 54

Description et mise en œuvre d'une méthode de commande permettant de traverser les singularités d'un manipulateur à six axes rotoides

Résumé : Ce rapport présente une méthode permettant de traverser "en douceur" les singularités géométriques d'un robot manipulateur rigide. Basée sur l'approche par fonctions de tâche de C. Samson ([5]), elle s'applique en fait à tout type de singularités. Cette méthode a déjà fait l'objet d'une application au cas d'un robot planaire redondant ([6]), et nous l'étendons ici au contrôle d'un robot à six degrés de liberté. Le rapport comporte deux parties principales, l'une dédiée à la théorie, l'autre à la mise en œuvre et aux expérimentations.
La méthode proposée se situe dans la classe des commandes dynamiques avec découplage et linéarisation par retour d'état. L'objectif de contrôle est défini par la minimisation d'une fonction scalaire positive qui exprime simultanément la poursuite d'une trajectoire dans l'espace des repères et la régularisation du problème. Ce dernier aspect est celui qui permet le passage au voisinage des singularités, grâce à l'utilisation d'une fonction du temps, image des vitesses articulaires. Cette méthode est une généralisation des approches usuellement présentées dans la littérature robotique.
La deuxième partie du rapport indique comment la loi de commande robuste proposée peut être spécifiée et testée en vraie grandeur à l'aide de l'environnement de programmation ORCCAD ([8]). On y précise la décomposition fonctionnelle de la commande, en prenant en compte les aspects temporels liés à l'implémentation, ainsi que la partie "événements discrets" de l'application. Enfin, on présente un ensemble complet de résultats concernant les singularités du poignet, du coude et de l'épaule.

Mots-clé : Robotique, Commande, Manipulateurs, Singularités

1 Introduction

Kinematic singularities are one of the major problems encountered when using robot manipulators with rotational joints. It is well known that, if not taking care enough in their neighborhood when using inverse differential kinematics, joint velocities and accelerations may dramatically increase, risking to damage to the robot and its environment. There are two main ways of coping with this problem. The first one is to generate open-loop trajectories which are far from the singularities; this is mainly done when the robot is redundant with respect to the task. The second idea is to try to cross *safely* the singular points, for example by transforming the ill-conditionned problem of jacobian inversion in a well-posed one, through a modification of the matrix to be inverted. This kind of regularization approach, already known under another form in numerical analysis ([1]), was introduced in robotics as a class of damped least-squares methods (see for example [3, 4]). The approach has then be applied and improved by several authors, like in [7]. However, to our knowledge, no work is reported in the literature about the practical implementation of such methods.

In fact, it appears that the concept of singularity may be understood in a wider sense than restricted to kinematic singularity. Related regularization methods may also be considered with a more general point of view, through the minimization of a cost function. Such an approach has been investigated in [5] by using the formalism of task functions. It has also be applied in the simple case of a planar redundant robot, as reported in [6]. The idea consists in setting the problem as the minimization of a quadratic cost function, which blends a trajectory tracking requirement in some task space and a regularizing term. This last term expresses a "safe" behavior of the robot in the neighborhood of a singularity, and depends on a time-varying function which may sometimes be interpreted as an image of the robot joint velocity. The relative weights of the two components of the cost function vary with respect to the proximity of the singularity. The goal of this paper is firstly to present this approach and to show how it provides a justification of the known methods already cited. In the second part of the paper, we describe specification and implementation issues through a real example. We use for that purpose the ORCCAD environment ([8, 9]) and we show how the method can be practically implemented. We present its functional decomposition, including temporal aspects. We also show what are the associated discrete events to handle, and how they can be controlled by an automatically generated automaton. Finally, results concerning wrist, elbow and shoulder singularity crossing for a 6 d-o-f robot are given.

2 A General Approach to Regularization in Robotics

2.1 Principles

Let q be the n-dimensional vector of the joint positions of a rigid robot. We would like to minimize at each time t the scalar cost function

$$h(q,t) = h_1(q,t) + h_2(q,t) \tag{1}$$

where h_1 is a positive function related to the task to be performed and h_2 is a positive regularizing function. The evolution of q is governed by the dynamics:

$$\Gamma = M(q)\ddot{q} + N(q, \dot{q}) \tag{2}$$

where Γ is a n-dimensional control vector.

The h function, assumed to be locally unimodal, is minimum when its gradient vanishes. We are therefore led to finding the control which will drive to zero the function $e(q,t)$ defined as

$$e = \frac{\partial h_1}{\partial q} + \frac{\partial h_2}{\partial q} \tag{3}$$

Then (cf [5]), a decoupling and feedback-linearizing control in the e-space is given by:

$$\Gamma = -kM(\frac{\partial e}{\partial q})^{-1}G(\mu De + \dot{e}) + N - M(\frac{\partial e}{\partial q})^{-1}f \tag{4}$$

with

$$\dot{e} = \frac{\partial e}{\partial q}\dot{q} + \frac{\partial e}{\partial t} \tag{5}$$

where k and μ are scalar positive gains, G and D constant positive symmetric matrices and f comes from

$$\ddot{e} = \frac{\partial e}{\partial q}\ddot{q} + f(q, \dot{q}, t) \tag{6}$$

From (4), we see that the task-jacobian matrix

$$\frac{\partial e}{\partial q} = \frac{\partial^2 h_1}{\partial q^2} + \frac{\partial^2 h_2}{\partial q^2} \tag{7}$$

has to be nonsingular. We will therefore have to choose the function h_2 such that this requirement (R1) be satisfied, *while disturbing the least possible the "main" task* h_1 (requirement R2).

2.2 A Class of h_2 functions

Like in [5], let us consider the set of functions

$$h_2(q,t) = \frac{1}{2}\lambda(t)(q - y(t)^T(q - y(t), \quad \lambda(t) > 0 \tag{8}$$

2.2.1 Choosing $\lambda(t)$

The scalar time function $\lambda(t)$ will be choosen so as to ensure that requirement R1 is satisfied. Using (1), (3), (8) and denoting $e_1 = \frac{\partial h_1}{\partial q}$, we have:

$$e = e_1 + \lambda(t)(q - y(t)) \tag{9}$$

and

$$\frac{\partial e}{\partial q} = \frac{\partial e_1}{\partial q} + \lambda(t) I_n \tag{10}$$

If $e_1(q,t)$ were any differentiable function, a sufficient condition for having $\frac{\partial e}{\partial q}$ regular would be

$$\lambda(t) > \| \frac{\partial e_1}{\partial q}(q(t),t) \| \tag{11}$$

Now, if e_1 is defined as the gradient of a unimodal function h_1, as above, the jacobian matrix of e_1, which is the hessian matrix of h_1, is positive. Therefore a sufficient condition for having $\frac{\partial e}{\partial q}$ nonsingular, and, more, positive (which is very useful from a control point of view), is

$$\nu_{min}(q(t),t) + \lambda(t) > 0 \tag{12}$$

where ν_{min} is the smallest eigenvalue of $\frac{\partial^2 h_1}{\partial q^2}$. Let us also note that $\lambda(t)$ can be also set equal to zero when we are sure to lie far from the task singularities.

2.2.2 Choosing $y(t)$

As suggested in [5], an interesting way of defining $y(t)$ such as to satisfy the requirement R2, is to consider that it is the output of the first-order filter:

$$\dot{y}(t) + \alpha(t) y(t) = \alpha(t) q(t) \tag{13}$$

with the initial condition $y(0) = q(0)$, and $\alpha(t)$ a positive scalar time function.

Intuitive Interpretation Let us choose $\alpha(t) = \alpha$, constant. Differentiating (13) leads to $\ddot{y} + \alpha \dot{y} = \alpha \dot{q}$, the steady state solution of which is $\dot{y} \rightarrow \dot{q}$. Replacing in (13) shows that $\alpha(q - y) \rightarrow \dot{q}$, which means that the second term of e in (9) looks like $\lambda(t)\alpha\dot{q}$. This means that, provided that the acceleration is not too important, the h_2 function defined in (8) represents a trend to minimize the joint velocity.

Choosing $\alpha(t)$ A reasonnable criterion for defining this gain function is to ensure that the velocity remains bounded. As proposed in [5], we can therefore set:

$$\alpha(t) = \frac{\beta_1}{1 + \beta_2 (\beta_3^2 + \| q(t) - y(t) \|^2)^{\frac{1}{2}}} \tag{14}$$

where all the β_is are positive. With that choice, we necessarily have:

$$0 < \alpha(t) < \frac{\beta_1}{1 + \beta_2 \beta_3} \quad ; \quad \| \dot{y}(t) \| < \frac{\beta_1}{\beta_2} \tag{15}$$

2.3 The Special Case of Trajectory Tracking in the Frame Space

2.3.1 Basic Issues

Let us set $n = 6$ and suppose that the task consists in tracking a desired trajectory in *position*, $x_d(t) \in R^3$, and in *attitude*, $r_d(t) \in SO(3)$. The associated rotation matrix is denoted as $R_d(t)$.

We can therefore take as a function to be minimized:

$$h_1(q,t) = \frac{1}{2} \parallel x(q) - x_d(t) \parallel^2 + h_1'(r(q), r_d(t)) \tag{16}$$

where h_1' represents some "distance" between the actual and desired attitudes. More precisely, let us define $r_e(q,t) = r(q) r_d^{-1}(t)$, with associated matrix R_e. Then, the geodesic distance between r_e and the identity is the angle θ, taken from $R_e = exp(\theta \tilde{u})$, where $\tilde{u}$ denotes the skew-symmetric matrix associated with the (unitary) vector u.

We can therefore choose:

$$h_1' = \frac{1}{2}\theta^2 \tag{17}$$

Using (16) and (17) in (9) leads to:

$$e(q,t) = \frac{\partial x}{\partial q}^T (q)(x(q) - x_d(t)) + \frac{\partial h_1'}{\partial q} + \lambda(t)(q - y(t)) \tag{18}$$

We have:

$$\frac{\partial h_1'}{\partial q} = \theta \frac{\partial \theta}{\partial q} = \theta \frac{\partial \theta}{\partial r} \frac{\partial r}{\partial q} \tag{19}$$

On the other hand, since θ is the geodesic distance to identity, it can be shown ([2]) that

$$\frac{d\theta}{dt} = < u(q,t), \omega(q,t) > = \frac{\partial \theta}{\partial r_e} \frac{dr_e}{dt} \tag{20}$$

where u is defined above and $\omega = \frac{dr_e}{dt}$ is the angular velocity. We therefore find $\frac{\partial \theta}{\partial r_e} = u$, and then:

$$\frac{\partial h_1'}{\partial q} = \frac{\partial r}{\partial q}^T (\theta u) \tag{21}$$

By denoting the robot Jacobian as $J = \begin{pmatrix} \frac{\partial x}{\partial q} \\ \frac{\partial r}{\partial q} \end{pmatrix}$, we have finally, using (21) and (18):

$$e(q,t) = J^T \begin{pmatrix} x(q) - x_d(t) \\ \theta u \end{pmatrix} + \lambda(t)(q - y(t)) \tag{22}$$

Now, in order to complete the definition of the control scheme (4), we have to give the expressions of $\frac{\partial e}{\partial q}$ and $\dot{e}$ or $\frac{\partial e}{\partial t}$. Starting from $R_e = R R_d^{-1}$, we have:

$$\frac{dR_e}{dt} = \tilde{\omega} R_e - R_e \tilde{\omega}_d \tag{23}$$

where $\tilde{\omega}$ is the actual angular velocity and $\tilde{\omega}_d$ the desired one. We may define $\tilde{\omega}_e$ (such that $\frac{dR_e}{dt} = \tilde{\omega}_e R_e$), as:

$$\tilde{\omega}_e = \tilde{\omega} - R_e \tilde{\omega}_d R_e^T \tag{24}$$

where $R_e \tilde{\omega}_d R_e^T$ is no more than $\tilde{\omega}_d$ expressed in the basis defined by R_e.

Now, we can obtain the needed derivatives of e:

$$\frac{\partial e}{\partial q} = J^T J + \lambda I_6 + \sum_{i=1}^{3} \left(\frac{\partial^2 x_i}{\partial q^2}(x - x_d)_i + \frac{\partial^2 (\theta u)_i}{\partial q^2}(\theta u)_i \right) \tag{25}$$

where the subscript i denotes the index of the vector component, and:

$$\dot{e} = J^T \begin{pmatrix} \dot{x} - \dot{x}_d \\ \tilde{\omega}_e \end{pmatrix} + \dot{J}^T \begin{pmatrix} x(q) - x_d(t) \\ \theta u \end{pmatrix} + \dot{\lambda}(q - y) + \lambda(\dot{q} - \dot{y}) \tag{26}$$

2.3.2 Parametrization and Approximation

When θ is small, we have $\theta \approx \sin\theta$. We can therefore replace equation (22) by:

$$e(q,t) = J'^T \begin{pmatrix} x(q) - x_d(t) \\ \sin\theta u \end{pmatrix} + \lambda(t)(q - y(t)) \tag{27}$$

with

$$J' = J \begin{pmatrix} I_3 & 0 \\ 0 & W(r) \end{pmatrix} \tag{28}$$

where $W(r) = \frac{1}{2}(tr(R)I_3 - R)$is the derivative of the parametrization, $\frac{\partial \sin\theta u}{\partial r}$.

The interest of this parametrization lies in the fact that it is very easy to compute by using the expression:

$$\sin\theta u = \frac{1}{2}(n \times n_d + s \times s_d + a \times a_d) \tag{29}$$

where $R = (n\ s\ a)$ and $R_d = (n_d\ s_d\ a_d)$. In that case, we can use in the control expression:

$$\frac{\partial e}{\partial t} = J'^T + \begin{pmatrix} -\dot{x}_d(t) \\ \frac{1}{2}(n \times \dot{n}_d + s \times \dot{s}_d + a \times \dot{a}_d) \end{pmatrix} + \dot{\lambda}(q - y) - \lambda \dot{y} \tag{30}$$

In practice, simplified expressions of the derivatives may be used in the control. If we consider that the trajectory error is small, we may neglect the two last terms in equation (25). Furthermore, since θ is small, $W(r)$ is positive and close to the identity. We may therefore finally use as a model of the jacobian matrix $\frac{\partial e}{\partial q}$ associated with (27) the simpler expression:

$$\hat{\frac{\partial e}{\partial q}} = J^T J + \lambda I_6 \tag{31}$$

which is the model used in most of the literature. It should be emphasized that this approximation is justified when the true jacobian is itself positive in a large enough domain. This comes from a stability analysis reported in [5].

Finally, it is also possible to simplify the expression of $\frac{\partial \hat{e}}{\partial t}$ by suppressing the two last terms in (30), provided that the velocities are small and that $\lambda(t)$ slowly varies. In any case, increasing the gain μ in (4) may allow to reduce the tracking error due to a non perfect precompensation of the task velocity.

3 Applications

3.1 Relations with Commonly Used Methods

Before describing how this approach can be specified, programmed, validated and implemented in a systematic way through a generic environmment, let us briefly relate the above developments to a relevant example of what is usually done in reported works in that area. In [7], the problem of coping with shoulder, elbow and wrist singularities for a 6 dof robot is addressed in a clever way. The distance to a singularity is evaluated by the conditioning of the robot jacobian matrix, through an estimation of its smallest singular value. This last is used to compute the value of λ in equation (31). The orientation tracking error is defined as in (29). We may also find in the paper an expression of the desired angular velocity looking like the one in (24), which is however not defined as the time derivative of the trajectory to be tracked. The reason is that the problem is not here set as a minimization and control one, but simply as the computation of some inverse differential kinematics. This is in fact the major difference with the framework we propose. Nevertheless, some other common ideas appear. For example, the variable weighting between "velocity" control and "trajectory tracking", characterized in our approach by $\lambda(t)$, is also achieved in the cited paper by assigning to the trajectory tracking a coefficient which varies with the smallest singular value. We can therefore consider that, in some sense, the reported work, like others, includes some of the features described above, provided that underlying approximations are made explicit. These lasts can be justified from our general description.

4 Experimentation under Orccad

It is (or it should be) well known that a robot control problem cannot be considered as solved when the theoretical analysis in continuous time is completed. It is really necessary to consider other issues, like discretization and numerical aspects, design of the discrete-event part of the problem, parallelism and task synchronization, etc... In the present case, these aspects are particularly relevant, at least for two main reasons: first, we deliberately move the robot around the singularities, therefore the *practical* conditioning of the problem have to be studied. Secondly, we have a control scheme which naturally expands in a set of communicating and parallel tasks; moreover, some of them, like the detection of singularity

approach and the tuning of the regularization terms handle discrete events which might influence the control behavior in some way. We therefore describe in this section how we have considered all these issues, critical from the implementation point of view.

4.1 The Orccad system

ORCCAD ([8]) is a development environment for the specification, the validation by simulation and by formal methods when possible, and the implementation of robotic applications. In the ORCCAD framework, an application is defined as the composition of a set of elementary robotic actions (Robot-tasks). Each can be seen as an event-driven action which merges algorithmical and behavioral aspects. Composition is obtained using logical and temporal operators determining so, in a predefined way, the evolution of the system in order to achieve the desired objective. Nominal executions as well as degraded executions are taken into account. The underlying models are those used in automatic control theory (ODEs) for the algorithmical parts and in computer science (DES) for the behavioral one. The originality of ORCCAD is that both aspects are closely considered at all the involved levels, from specification to real-time implementation.

Since the work presented here concerns the design and the analysis of a single task, i.e the tracking of a reference trajectory in the frame space, we focus the following description on the concept of Robot-task.

A *Robot-task* (RT) is formally defined as the entire parametrized specification of:

- an elementary control law, i.e. the activation of a control scheme structurally invariant along the task duration,
- a logical behavior associated with a set of signals (events) which may occur just before, during and just after the task execution.

The control law specification is obtained by selecting a set of functions, models and parameters appearing in the analytic expression, in continuous time, of the torques to be applied to the actuators in order to achieve the desired physical action. In the example of a rigid robot manipulator, accordingly to equation (4), functions like the task function $e(q,t)$, models like $\widehat{M}$ and parameters like the scalar gains k and μ have to be be specified. Besides, the logical behavior is given by setting the events to be considered and their processing. In order to facilitate and automatize this last aspect, rarely considered in the automatic control community, the events and the associated treatments in ORCCAD are typed and belong to well-defined categories. These are,

- the *pre-conditions*: their occurrence is required for starting the servoing task. They can be pure synchronization signals, or signals emitted when some task initial requirements are satisfied as well as signals issued from environment sensors. For instance, the condition $\| e(q_0,0) \| < \epsilon$ is necessary to ensure a good behavior of the control law. Another example is a grasping action, where the detection of part presence is required before starting an action.

- the *exceptions*: they are generated during the execution of the servoing task and indicate a failure detection. The exception processing is structured as follows:

  type 1: the reaction to the exception consists only in asking for the modification of the value of at least one parameter in the control law. After its termination, a newly occured exception can be processed. Here, this type of treatment is used, as detailed in the next section, to handle the situation where the entrance into a singular region is detected.

  type 2: the exception requires the activation of a new RT. The current one is therefore killed, while the causes of the dysfunction are reported to the application controller. As mentioned at the beginning of this section, the recovering process to be activated is known, having been specified during the application design stage. Let us cite as an example the switching to a reconfiguration task when reaching a joint limit.

  type 3: the exception is considered as fatal. The overall application is stopped and the robotic system must be driven to a safe position. For example, an excessive increase of the tracking error allows to infer an actuator failure or a collision.

- the *post-conditions*: they are signals, often related to the environment, handled as conditions for a normal termination of the RT.

Finally, the RT specification ends by the explicitation of implementation aspects, gathered under the name of *temporal properties*, i.e. discretization of time, duration of computations, communication and synchronization protocols between the involved processes. This is done by implementing each RT in terms of a network of communicating real-time computing tasks, called *Module-tasks*, which each implement an elemental part of the control law and the related behavioral components.

The ORCCAD system actually provides the necessary data structures allowing an effective RT specification. The diversity and the large number of possible choices of the required elements makes it a not trivial operation. To enhance guidance and coherence tests the RT is modeled using an object-oriented approach handling several hierarchies of classes, related to the control scheme, to the logical behavior and to the involved physical entities. Accordingly to this model, a RT is completely defined by the instantiation of a set of objects, leaves of these hierarchies. Each object is characterized by two type of attributes: the *functional* ones, relative to the continuous time specification, and the *operational* ones characterizing the temporal properties. The associated methods interpret the models informations at various levels, down to real-time code generation.

In fact ORCCAD proposes a general methodology for the specification, the validation and the implementation of robot tasks, which will be illustrated in the next section on the example of singularity handling. The design process can be summarized as follows: in a first step, the specification in continuous time, which characterizes the task from the automatic control point of view, is established. Functional only attributes of the concerned objects are valuated in this stage. Then, it can be extended and take into account implementation issues; during this phase the object instantiation is completed by valuating their attributes

related to the temporal properties. This specification is translated to a simulation program in order to allow the user to update the model choices and to tune the control parameters as well as the temporal ones. The final stage of the design consists in the generation of the downloading code to be executed by the robot system.

TheORCCAD software environment includes a graphical HMI which makes easy the RT specification, by representing it in the form of a block-diagram. Dedicated interfaces are provided for editing functional and operational attributes, by simply clicking on meaningful spaces in the RT graphical representation. An automatic code generation, for both simulation and execution purposes, is also provided in relation with the SIMPARC ([10]) simulation tool and the VXWORKS real time operating system. In addition, the ESTEREL ([11]) synchronous language is used for the implementation of the reactive components of the RT. We must note that, from the end-user point of view, this environment allows to perform the specification, simulation and execution phases of the RT design process through a single graphical development environment.

In the next section, we describe how the task addressed in this paper can be designed and tested using the ORCCAD methodology and the associated programming environment.

4.2 Trajectory Tracking in the Frame Space with Singularity Handling: Action Specification

4.2.1 Description

The task we consider to illustrate our approach for handling the kinematic singularities of a robot manipulator consists in tracking a reference trajectory defined in the SE(3) space. The robotic system to be used is the AID-V5, a six-revolute joint manipulator. Its Denavit-Hartenberg (D-H) parameters and the associated joint ranges are reported in table 1. A schema of the manipulator is shown figure 1.

Let us note that, before physically moving the manipulator, the correctness of the initial conditions and of the initializations have to be be checked; a signal emitted by the user must also be waited for before starting the task. The action must be aborted when entrance in the joint limits area is detected, while the passage through singular regions must be ensured. The duration of the servoing task is that of the reference trajectory.

4.2.2 Specification in Continuous Time

On the basis of previous data and requirements, a set of objects, proposed at the RT modelling level, can be instantiated by valuation of their functional attributes. The resulting specification characterizes the action from a continuous-time point of view, i.e independently of time discretization and other implementation-related issues, which are precised at the next step of the specification process.

In the following, we describe the functionality and the main characteristics of the considered objects, related to the servoing task. A graphical representation of the RT, as designed using the RT editor of the ORCCAD system, is given figure 2.

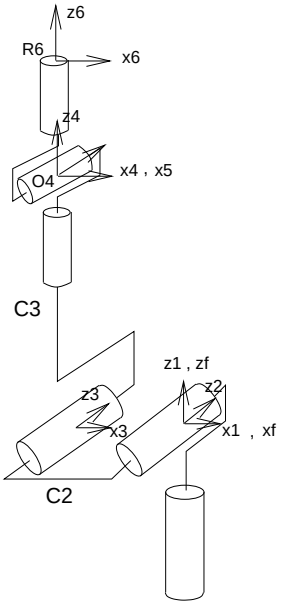


Figure 1: The Structure of the AID-V5 at the Configuration $\theta = [0.0 \quad -1.57 \quad 1.57 \quad 0.0 \quad 0.0 \quad 0.0]\,[rad]$.

Link	σ	α	d	θ	r	*Range limits*
1	0	0	0	q_1	0	$-3\pi/4\ ;\ 3\pi/4$
2	0	$\pi/2$	0	q_2	0	$-3\pi/4\ ;\ 3\pi/4$
3	0	0	0.26	q_3	0	$-3\pi/4\ ;\ 3\pi/4$
4	0	$-\pi/2$	0	q_4	0.40	$-\pi\ ;\ \pi$
5	0	$\pi/2$	0	q_5	0	$-2\pi/3\ ;\ 2\pi/3$
6	0	$-\pi/2$	0	q_6	0.13	$-\pi\ ;\ \pi$

Table 1: Denavit-Hartenberg Parameters and Range Limits of the AID-V5 Manipulator

TG_SE3 is aimed to generate on line the reference trajectory $R_d(t)$ in the SE(3) space. The choices of initial $R_{init} = [A_{init}, P_{init}]$ and final $R_{fin} = [A_{fin}, P_{fin}]$ positions, possibly of the intermediate passage points, and of the trajectory duration T are left to the designer. When intermediate points are not specified, the velocity and acceleration profiles of the trajectory are fixed by choosing the $r(t)$ function in the following expression:

$$P_d(t) = P_{init} + r(t)[P_{fin} - P_{init}] \quad \text{and} \quad A_d(t) = A_{init}A(u, r(t)\Theta), \quad t \in [0, T] \tag{32}$$

with u the vector along the axis of the rotation and Θ the rotation angle. In section 5 we present several reference trajectories of that kind designed in order to pass throw shoulder, elbow and wrist singularities.

AID_PR models, for simulation purposes, the dynamics of the AID-V5 manipulator using the state representation

$$\dot{X} = f(X, U, t), \quad Y = g(X, U), \quad X = [q_i, \dot{q_i}, i = 1, \ldots, 6]$$

where f and g functions are provided by the user.

KIN computes the current attitude of a frame linked to the end effector. The user has to specify the geometric characteristics of the robot (dofs and D-H parameters) and to select the expression frame and the frame of interest. In our case, the values of table 1 are used. We chose as a frame of interest the end effector frame R_6, and as an expression frame the fixed frame R_F (fig. 1).

DIF computes the robot jacobian J. For its instantiation, the same choices as for the KIN object have to be done.

TF_SE3 is aimed to compute the task function $e(q,t)$ following (27). Actually, the computation of $\lambda(t)$ guarantees its smooth evolution despite changes of the boolean variable indicating when the manipulator lies inside or outside a singular region. So, a change of the indicator from 0 to 1 at the instant t_k implies

$$\lambda(t) = \begin{cases} \lambda_{t_k} + \xi(t)(\lambda_{max} - \lambda_{t_k}), & t < T_\lambda \\ \lambda_{max}, & t > T_\lambda \end{cases} \tag{33}$$

and, reversely, the passage of the indicator from 1 to 0 at t_k implies

$$\lambda(t) = \begin{cases} \lambda_{t_k} - \xi(t)\lambda_{t_k}, & t < T_\lambda \\ 0, & t > T_\lambda \end{cases}$$

where $\xi(t)$ is chosen to be a second order polynomial function ensuring $\lambda(t)$ to be C^2. λ_{max} and T_λ are parameters tuned by the user. The information of entering a singular region is broadcasted by the discrete event controller, modelled by the object RTA presented later, as a response to the exception detected by the observer OBS_SING.

$y(t)$ is computed following (13). To ensure the boundedness of the filtered velocity we have chosen

$$\alpha(t) = \begin{cases} \frac{v_{max}}{\| q_{t_k} - y(t_{k-1}) \|}, & \text{when } \| q_{t_k} - y(t_{k-1}) \| > v_{max} \\ 1, & \text{otherwise} \end{cases}$$

The designer has also to determine the type of parametrization of the attitude error to be used, and the value of v_{max} used in $\alpha(t)$ expression. In our case the axis/angle parametrization is chosen and v_{max} is fixed to 0.0015.

JD_SE3 computes the task jacobian: its functionality is to produce $\frac{\partial e}{\partial q}$. The possibility of chosing approximated expressions is also offered to the designer: the transpose of the robot jacobian, the identity matrix, ... Here, the model given by (31) is used. $\lambda(t)$ is computed following 33.

JI_SE3 and **TD_SE3** are aimed to compute the models of the inverse of the task jacobian and its time derivative respectively. In the present specification, the inverse of (31) is chosen for the first one, and the simplified expression of (30) for the second.

CO_SE3 has the responsibility to produce the torque Γ to be applied to the actuators. It is computed following (4). The RT designer should valuate all the scalar and matricial gains, k, μ, G and D. Here we have $k = 15$, $\mu = 17$, $G = I$ and $D = I$.

The Discrete Event aspects of the RT are specified by instantiating four objects, three of them are called *observers*:

OBS_PREC detects the user's command allowing to start the action, and sends the corresponding event, "*start*", to the discrete event controller.

OBS_SING detects the entrance in the singular regions, and reports that to the discrete event controller by emitting the event "*sing*". The designer has to choose the type of the measure of proximity to be used, and fixes the related thresholds. Here, among the observation of the singular values of the robot jacobian and that of its determinant, we chose the determinant. In that case, the threshold det_s is taken in the interval [0.001 , 0.025] as seen in section 5.

OBS_BUT is aimed to detect the approach of joint limits. It reports this occurence to the discrete event controller by emitting the event "*joint_lim*". The designer has to define the joints to be observed, their working ranges and the related thresholds. In our case, all the joints are considered, and the values given in the table 1 are used. The threshold is set equal to 0.001.

ATR_SE3 imposes to the RT the desired logical behavior, driven by the observed input events and the emitted output ones, and described by a finite state automaton. The designer has to specify the type of processing associated with the considered exceptions. In the present example, the T3 type (fatal) is associated with the "*joint_lim*" event while a T1 (local) processing corresponds to the "*sing*" one.

4.2.3 Time Constrained Specification

The passage from the continuous-time specification like the previous one to a description which takes into account the implementation aspects is achieved by specifying the so-called *temporal properties*: sampling periods, duration of computations, communications and synchronizations protocols between the processes and, finally localization of the data processing codes.

At this level of specification, the basic entity to be handled is called a Module-task: it is a real-time task, usually periodic, used to implement a functional module of the RT. Since

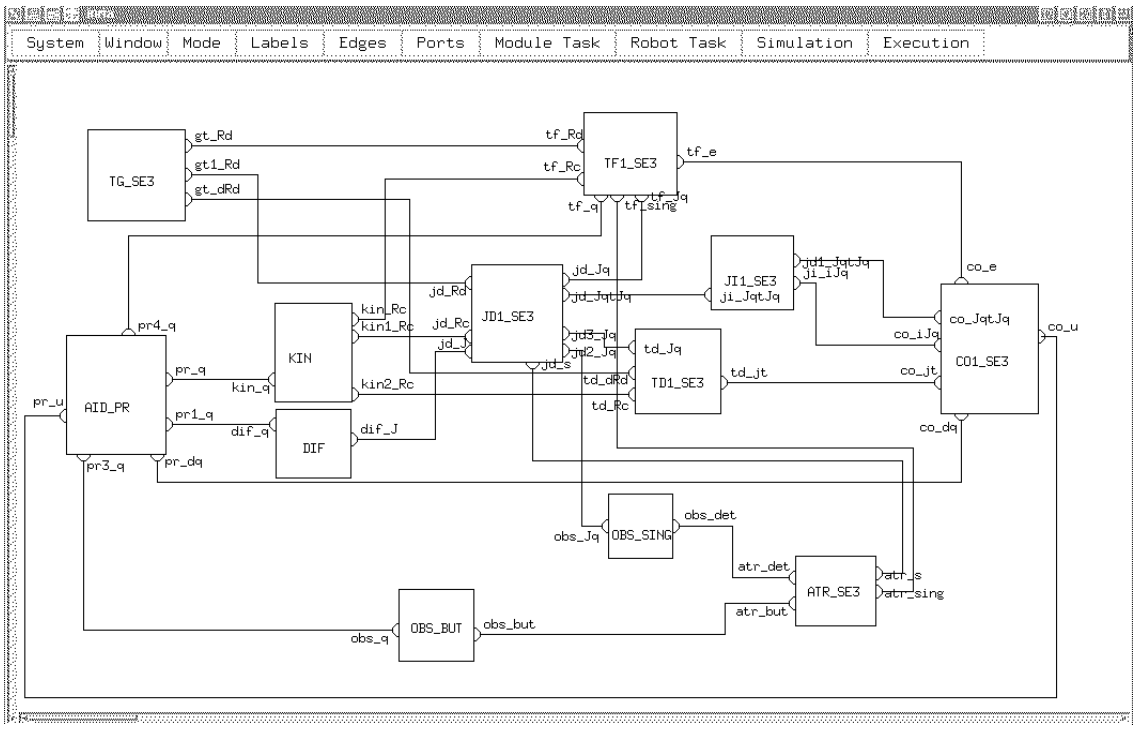


Figure 2: Graphical representation of the Robot-task: Trajectory Tracking in the Frame Space

MTs may, possibly, be distributed on a multi-processor architecture, they communicate using typed ports and specific synchronization mechanisms. Each MT owns one input (resp. output) port for every data it consumes (resp. produces).

In practice, the designer has, at this step, to add temporal properties to the already instantiated functional objects. The ORCCAD software environment proposes dedicated editors for their effective valuation.

In order to illustrate this part of the specification process, let us give some examples taken from the considered application: the TG_SE3 module has three output ports, *gt_Rd*, *gt1_Rd* and *gt_dRd*, which send, at every time period, the desired trajectory R_d to the TF and JD modules, and its differential *dRd* to the TD module (fig. 2). Since no input data are used by the, TG_SE3 module, no input port is needed.

Three different types of synchronization are associated with the created communications: *Asynchronous/Synchronous* between CO and PR_AID: the control outputs sent to the actuators are the last computed ones; *Event* for the communications between the observers and the ATR module, ensuring the safe and ordered transmission of the events; *Asynchronous/Asynchronous* in all other cases.

Finally, the sampling periods are determined for every module. For example, it is fixed at 15ms for TG_SE3; its execution duration is estimated at 90μs. For the CO module (control), which has clearly to be the most frequently updated one, the period is fixed to 10ms.

5 Experimental Results

Owing to the ORCCAD environment, and starting from the RT previously specified, a large set of experiments can be easily conducted by only modifying the functional elements in relation with the desired experimentation. In our case, where we address the problem of crossing kinematic singularities, we can for example consider several cases of singularities by simply specifying different reference trajectories for the same tracking task. For that reason, we will only detail these aspects in the presented results, since all the other issues remain unchanged. Namely, they are: the reference trajectory to be tracked, the threshold det_s which determines if the current configuration lies inside or outside a singular region, and, finally, the λ_{max} and T_λ parameters allowing to give to $\lambda(t)$ a suitable shape in the neighborhood of the singularity. We analyze in the sequel the shoulder, wrist and elbow singularities of the AID manipulator and, for each one, we present significant case studies.

5.1 Shoulder Singularities

5.1.1 Analysis

It is well known that, for a robot structure like the AID-V5, shoulder singularities occur when the origin O_4 of the frame associated with the fourth joint is on the z_1 axis. In this case the only admissible velocities of O_4 are in the plane defined by the links C_2 and C_3 (fig. 1).

These singular positions can be determined from the jacobian matrix [1] of the O_4 position function:

$$ {J_0}^4 = \begin{bmatrix} s1 \cdot (r_4 \cdot s23 - d_3 \cdot c2) & -c1 \cdot (d_3 \cdot s2 + r_4 \cdot c23) & -r_4 \cdot c1 \cdot c23 \\ c1 \cdot (d_3 \cdot c2 - r_4 \cdot s23) & -s1 \cdot (d_3 \cdot s2 + r_4 \cdot c23) & -r_4 \cdot s1 \cdot c23 \\ 0 & -r_4 \cdot s23 + d_3 \cdot c2 & -r_4 \cdot s23 \end{bmatrix} $$

where r_4 and d_3 refers to the D-H parameters.

Analyzing its determinant

$$ det({J_0}^4) = (r_4 \cdot sin(q_2 + q_3) - d_3 \cdot cos(q_2)) \cdot cos(q_3) $$

we obtain that singular points are given by the relations

$$ r_4 \cdot sin(q_2 + q_3) - d_3 \cdot cos(q_2) = 0 \tag{34} $$

and

$$ q_3 = 0 \;\; or \;\; q_3 = \pm\pi \tag{35} $$

The set of points given by (34) correspond to the situation discussed above, i.e O_4 positioned on the z_1 axis. Equation (35) determines the elbow singularities; the present analysis is also valid for them and we will not repeat it in section 5.3, where only a particular case will be presented.

As explained in [5], several cases of task singularities may occur, depending on the desired trajectory. The simplest one is when an isolated singular point is encountered while tracking this trajectory. This occurs when the desired trajectory crosses a set of singular points 'transversally', like the trajectories shown in figure 3a. Two subcases must be considered:

• the first one, illustrated by the trajectory (a), is characterized by the fact that, when O_4 leaves its singular position, it stays in the same plane (P_1); therefore, the needed velocity is in the reachable space,

• the second corresponds to the trajectory (b), where the desired velocity has a non zero component in the non-reachable space; O_4, when leaving the singular position, moves in fact in a different plane (P_2).

Besides, very difficult situations occur when the reference trajectory 'stays' on the singular points. This is the case of the trajectory (c) figure 3b: it reaches the z_1 axis while moving in the plane (P_1) and leaves it by moving in the plane (P_2). Because q_1 is undetermined as long as O_4 is on the z_1 axis, the value of q_1 corresponding to the motion in the plane (P_2) is unpredictable. If q_1 does not have the correct value when the reference trajectory leaves the z_1 axis, it has to change instantaneously, which is physically impossible.

These different cases are studied and simulation results are presented in the following sections.

[1] In its expression si denotes $sin(q_i)$, ci denotes $cos(q_i)$, sij denotes $sin(q_i + q_j)$ and finally cij denotes $cos(q_i + q_j)$

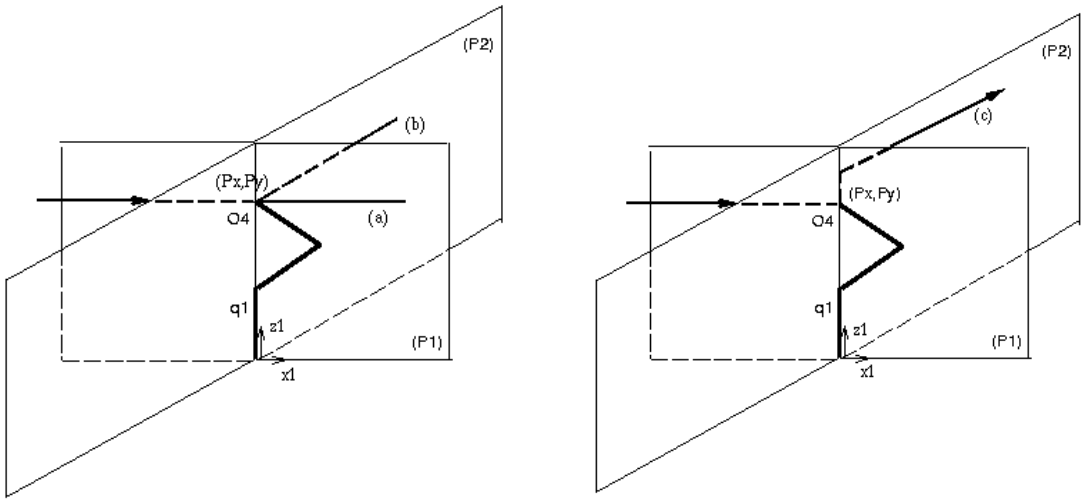


Figure 3: Crossing a) 'Transversally' b) 'Staying on' Shoulder Singularities

5.1.2 Case Study 1: 'Transversally' Crossing

Trajectory 1: A reference trajectory in the cartesian space looking like the (a) of figure 3a is specified by

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & -0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, Rd_{fin} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.39 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, \quad T = 3sec$$

It is computed by using equations (32), with a third order interpolation function ensuring continuity in position, velocity and acceleration. The initial position is $[0.0\ \ -1.57\ \ 1.57\ \ 0.0\ \ -1.57\ \ 0.0]\ [rad]$ (fig. 4a).

Note that the origins of the successive frames $R_d(t), t \in [0, T]$ lies on the same plane, their x coordinate varying from -0.13m to 0.39m, while their orientation is constant. Knowing the AID-V5 geometry, it is clear that imposing to its last frame R_6 this reference trajectory ensures that O_4 crosses 'transversally' the singular positions.

To obtain a smooth evolution, the det_s and λ_{max} values are fixed at 0.001 , and T_λ at 100ms. In fact, the smallest value of λ ensuring the inversibility of $\frac{\partial e}{\partial q}$ would be sufficient to cross smoothly the singularity. Plots in figure 4b show the resulting evolution of the jacobian determinant $det(J)$ and of $\lambda(t)$ during a simulation session. The entrance in the singular region detected at 1.4s sets to $\lambda(t)$ the evolution described by (33). We can remark that the time at which the determinant vanishes is the one where the x and y coordinates of O_4 are equal to zero (fig. 4c), i.e O_4 lies on z_1 axis. Finally, the desired trajectory is followed

without any additional error due to the smoothing method. Figure 4d shows the evolution of the x coordinate of the origin O_6 of the last frame R_6 varying smoothly between -0.13m and 0.39m, as expected.

Trajectory 2: Let us now present the results of the proposed method in the case where the reference trajectory is such that, when O_4 leaves the singular position, the asked velocity has a non zero component in the non reachable space.

The initial point of the reference trajectory (fig. 5a) is chosen as previously:

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & -0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}$$

while the final one is:

$$Rd_{fin} = \begin{bmatrix} 0.0 & -1.0 & 0.0 & 0.13 \\ 0.0 & 0.0 & 1.0 & 0.399 \\ -1.0 & 0.0 & 0.0 & 0.449 \end{bmatrix}$$

Moreover, the trajectory imposes the passage by the intermediary point

$$Rd_{int} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}$$

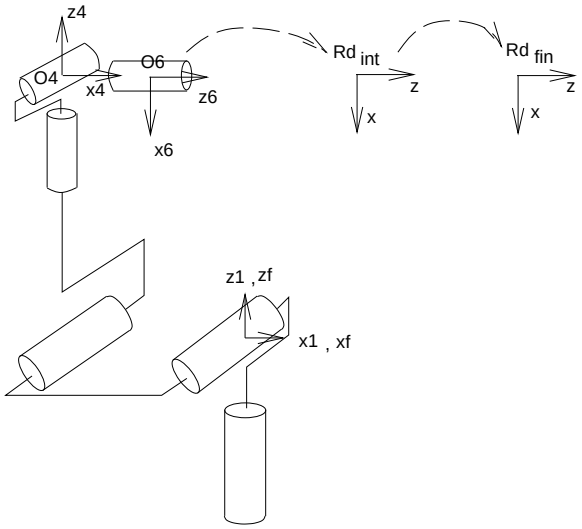
The initial position is, as for the trajectory 1, $[0.0 \ \ -1.57 \ \ 1.57 \ \ 0.0 \ \ -1.57 \ \ 0.0]$ $[rad]$.

Let us emphasize that the position component of the reference trajectory belongs successively to two different planes. We can indeed see that the first part of the trajectory is characterized by the evolution of the x coordinate of the frame origin, the y one remaining constant, while reversely, in the second part, the y coordinate increases. The total trajectory duration is of 3s. The Interpolation was realized using fourth degree splines ensuring continuity in position, velocity and acceleration at the intermediary point. Note that Rd_{int} corresponds to the R_6 attitude which imposes to O_4 to be on a singular position. Therefore, the total trajectory is an example of the (b) trajectory, figure 3a.

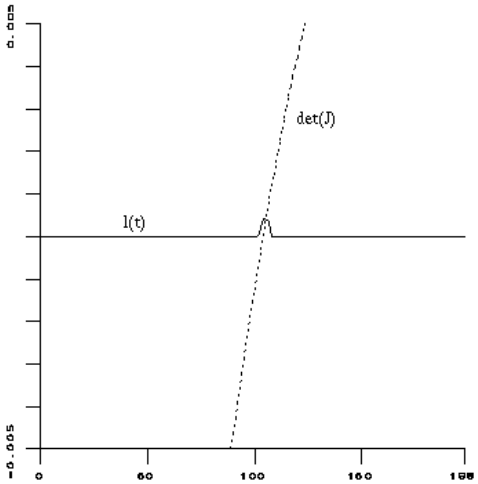
When the singularity treatment is not activated ($det_s = 0.0$), the expected behavior appears on the results. The singular configuration is reached at 1.5s, where $det(J) = 0.0$ (fig. 5b). The attempt to follow a non attainable direction causes excessive torques values, trying to instantaneously achieve unfeasible joint velocities and accelerations on the 1, 4 (fig. 5) and 6 axes , i.e. the axes related to the motion in the second plane.

The same experiment was conducted with the singularity treatment being active: the det_s value is set at 0.025, the λ_{max} at 0.001 and the T_λ at 600ms. The approach of the singular region is detected at 0.6s, when $\| \ det(J) \ \| < det_s$, which imposes at $\lambda(t)$ the evolution described by (33) and plotted in figure 6a. The x and y coordinates of O_4 clearly indicate its passage on the z_1 axis (point A, figure 6b).

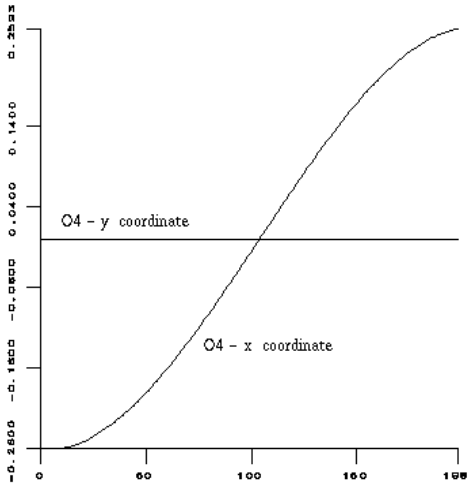
It can be seen that the control outputs and the joint velocities stay at an acceptable level, with a maximum peak value of approximately 2[rad/s], as shown figure 6d, for the



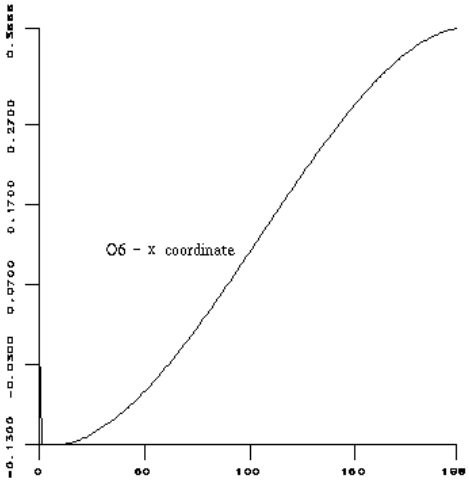
a) AID Initial Configuration



b) Determinant of the Robot Jacobian and $\lambda(t)$

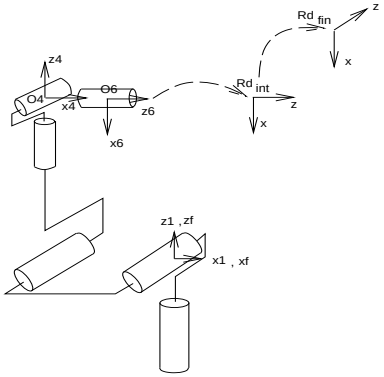


c) x and y Coordinates of O_4

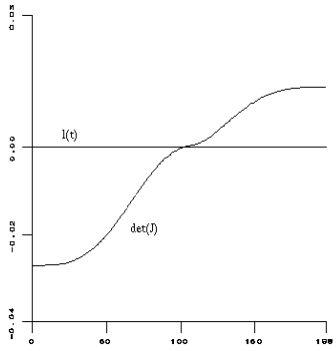


d) x Coordinate of O_6

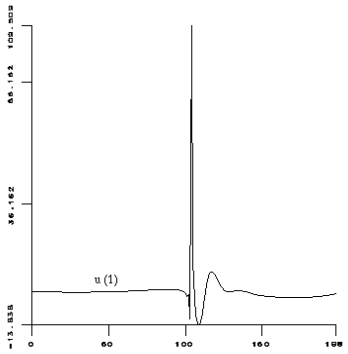
Figure 4: 'Transversally' Crossing Shoulder Singularities With the Smoothing Process (Trajectory 1)



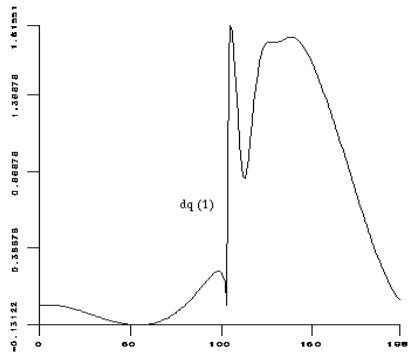
a) AID Initial Configuration



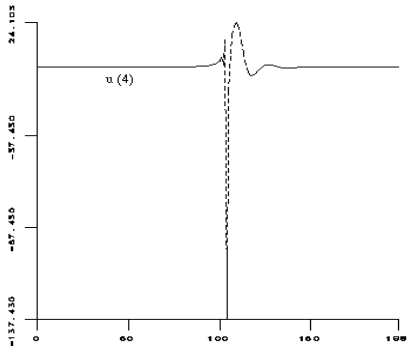
b) Determinant of the Robot Jacobian



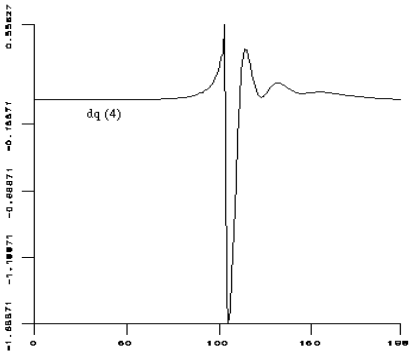
c) Torques Applied on Axis 1



d) Joint Velocities of Axis 1

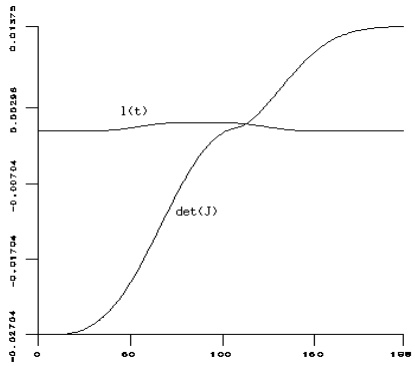


e) Torques Applied on Axis 4

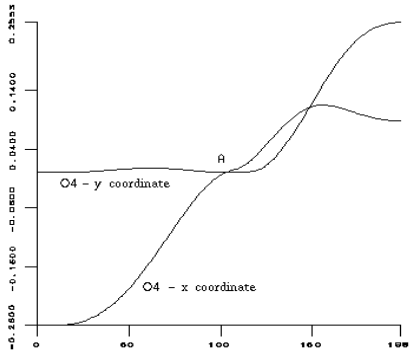


f) Joint Velocities of Axis 4

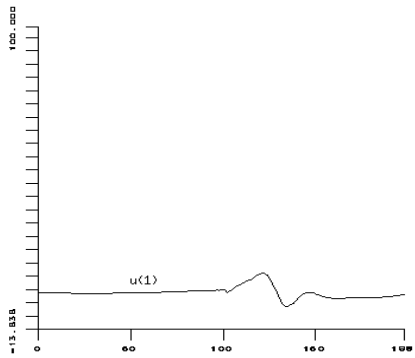
Figure 5: 'Transversally' Crossing Shoulder Singularities Without Singularity Processing (trajectory 2)



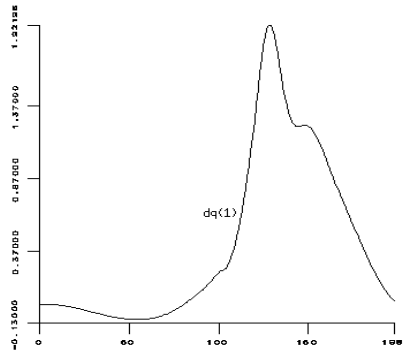
a) Determinant of the Robot Jacobian and $\lambda(t)$



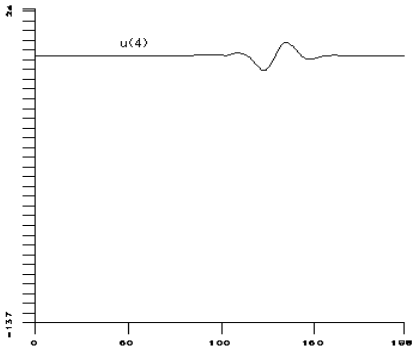
b) x - y Coordinates of O_4



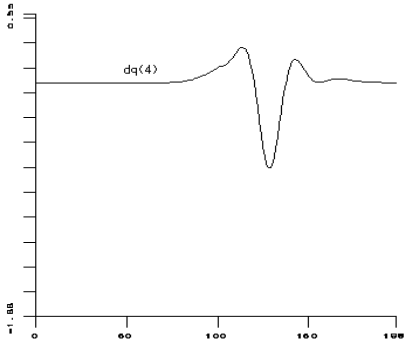
c) Torques on Axis 1



d) Joint Velocities of axis 1



e) Torques of Axis 4



f) Joint Velocities of Axis 4

Figure 6: 'Transversally' Crossing Shoulder Singularities with Singularity Processing (Trajectory 2)

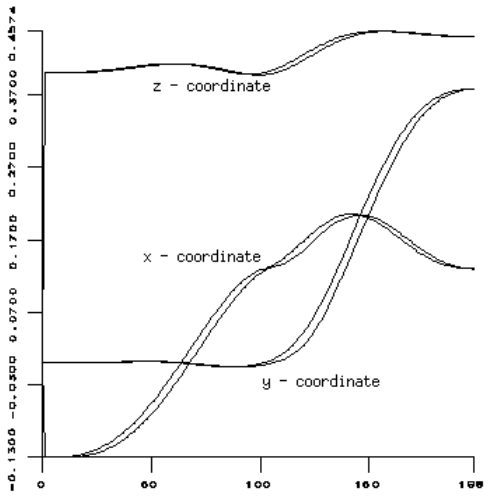


Figure 7: Reference and Current Position of O_6

first axis and of -0.4[rad/s] (fig. 6f) for the second one, plotted with the same scale as in the previous results.

Furthermore, a smooth trajectory tracking was obtained, as shown figure 7.

5.1.3 Case Study 2: 'Staying On' the Singularity

In this case, we consider a situation of type (c), figure 3b, where the reference trajectory is such that O_4 reaches a singular position while moving in a plane, follows the z_1 axis during a prescribed time and leaves it by moving inside a different plane.

A reference trajectory having the required characteristics can be specified by

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & -0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, Rd_{fin} = \begin{bmatrix} 0.0 & -1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.39 \\ -1.0 & 0.0 & 0.0 & 0.45 \end{bmatrix}$$

and we impose the passage by two intermediary points

$$Rd^1_{int} = \begin{bmatrix} -0.02 & 0.0 & 0.99 & 0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -0.99 & 0.0 & -0.02 & 0.37 \end{bmatrix}, Rd^2_{int} = \begin{bmatrix} 0.12 & 0.0 & 0.99 & 0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -0.99 & 0.0 & 0.12 & 0.55 \end{bmatrix}$$

The three sections of the trajectory defined in that way are interpolated using fourth degree splines for the first and final parts, and third degree for the second one ensuring continuity in position, velocity and acceleration on the intermediary points. The time duration of each section is 1.5 s.

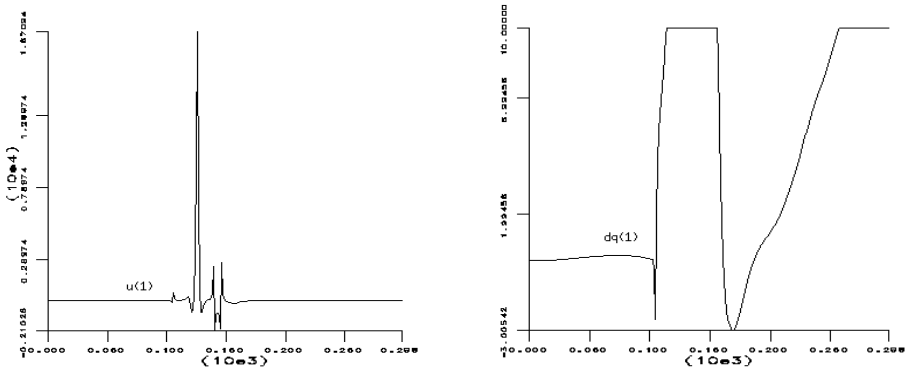
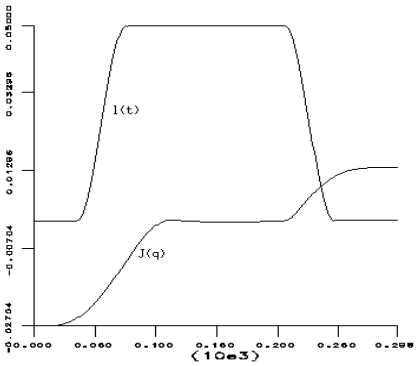


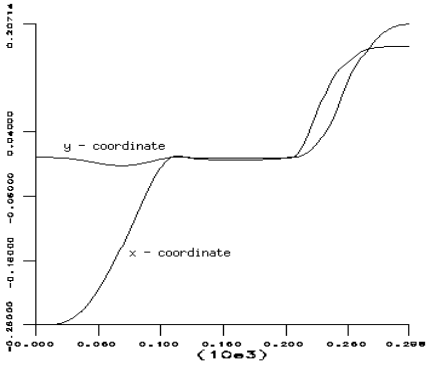
Figure 8: 'Staying on' Shoulder Singularities Without Singularity Processing (a) Torques (b) Joint Velocities on Axis 1

If the singularity processing is not provided ($det_s = 0.0$) the analysis of the simulation results clearly shows excessive torque values and infeasible joint velocities and accelerations when entering and staying in the singular region (fig. 8).

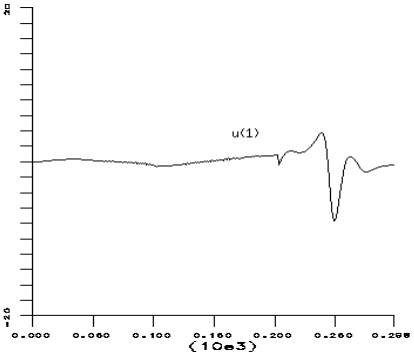
Besides, by activating the treatment and fixing det_s at 0.025, λ_{max} at 0.05 and T_λ at 600ms a smooth crossing is obtained. The entrance in the singular region detected at 0.6s leads for $\lambda(t)$ to the evolution plotted in figure 9a. The x and y coordinates of O_4 evolve as shown figure 9b; we can verify that, during the second part of the trajectory, they are, approximately, equal to zero, meaning that the robot stays on the singular configuration between 1.5s and 3s, while the z coordinate increases. The control outputs varies as shown figure 9. We can also see that, even in this very difficult example, the joint velocities remain always feasible. Furthermore, the resulting trajectory tracking is satisfactory, as shown fig. 10.



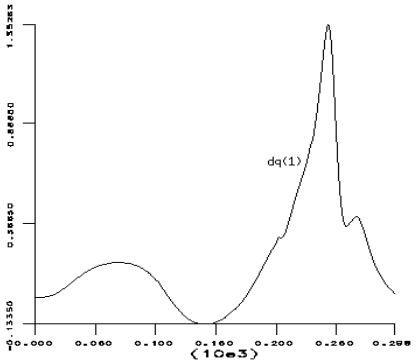
a) Determinant of the Robot Jacobian and $\lambda(t)$



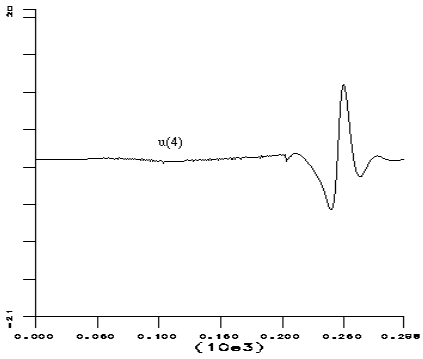
b) x - y Coordinates of O_4



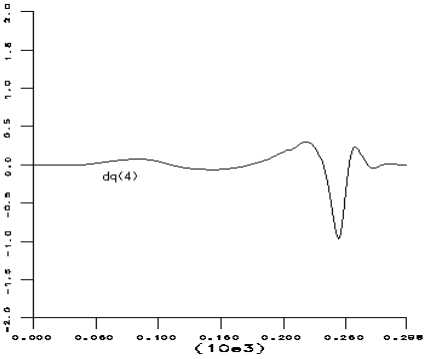
c) Torques on Axis 1



d) Joint Velocity of Axis 1



e) Torques on axis 4



f) Joint Velocity of Axis 4

Figure 9: 'Staying on' Shoulder Singularities with Singularity Processing (trajectory 3)

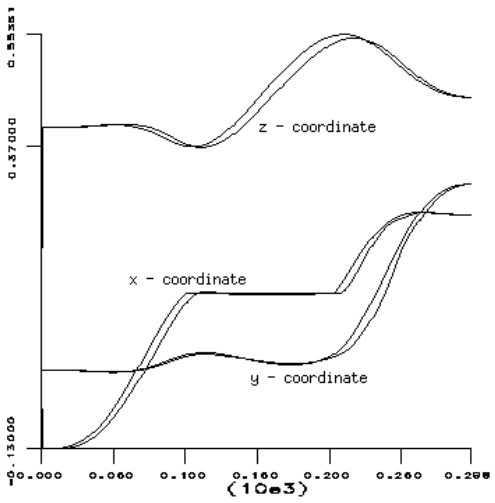


Figure 10: Reference and Current Positions of O_6

5.2 Wrist Singularities

5.2.1 Analysis

The AID-V5 manipulator is equipped with the common spherical wrist (fig. 11) involving the three Euler angles. Since it is a mechanical realization of the Euler chart, it necessarily has singular points which constitute the wrist singularities of the AID-V5. In order to analyze them, let us consider the wrist independently, and compute the angular velocity of O_6 in the basis associated with O_4. It is given by

$$\delta_6 = \sum_{k=4}^{6} [\bar{\sigma_k} a_k] \dot{q_k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \dot{q}_4 + \begin{pmatrix} sin(q_5) \\ -cos(q_5) \\ 0 \end{pmatrix} \dot{q}_5 + \begin{pmatrix} -cos(q_5)sin(q_6) \\ -sin(q_5)sin(q_6) \\ cos(q_6) \end{pmatrix} \dot{q}_6$$

where σ_k refers to the D-H parameters and a_k is the unit vector of the z axis of the joint k. We have therefore the jacobian matrix

$$J_R^w = \begin{pmatrix} 0 & sin(q_5) & -cos(q_5)sin(q_6) \\ 0 & -cos(q_5) & -sin(q_5)sin(q_6) \\ 1 & 0 & cos(q_6) \end{pmatrix}$$

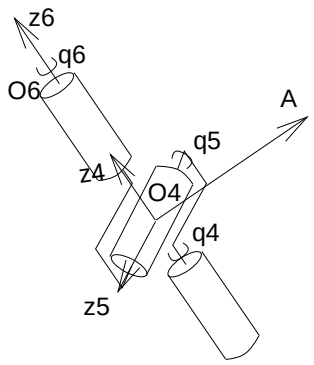


Figure 11: Wrist of the AID-V5

and its determinant is $det(J_R^w) = -sin(q_5)$. The singular points are given by $q_5 = 0$ or $q_5 = \pi$. At a singular point the jacobian matrix is

$$J_R^w = \begin{pmatrix} 0 & sin(q_5) & 0 \\ 0 & -cos(q_5) & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

indicating that no angular velocity along $[cos(q_5)\ \ sin(q_5)\ \ 0]^T$ is attainable.
In particular, in these configurations of the wrist, the AID-V5 manipulator looses one of its three degrees of freedom in orientation: the joints four and six produce the same result. Any attempt to rotate about the axis (A) (fig. 11) in the space perpendicular to both joint axis four and five causes O_4 to rotate quickly in one direction and O_6 in the opposite direction, so as to align the joint axis 5 perpendicularly to the specified motion. We will observe such a behavior later in the simulation results.

Situations analog to those illustrated in figure 3 and analyzed in section 5.1.1, dedicated to shoulder singularities, occur also in the case of the wrist. The wrist singularity region is crossed 'transversally' when the projection of the angular velocity of the desired trajectory onto $[cos(q_5)\ \ sin(q_5)\ \ 0]^T$ is zero at the time t_s where the robot meets the singularity. Besides, the most difficult situations take place when this projection is non zero at t_s or during a time interval.

Like in the case of shoulder singularities, we present in the following the results obtained when the reference trajectory is 'transverse' or 'stays' on the wrist singularities.

5.2.2 Case study 1: 'Transversal Crossing'

Trajectory 1: The reference trajectory is specified in the cartesian space by

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & -1.0 & -0.39 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, Rd_{fin} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & -0.13 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, \; T = 3sec$$

The overall trajectory is computed by imposing the intermediary point

$$Rd_{int} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -0.26 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.53 \end{bmatrix}$$

The interpolation uses fourth degree splines ensuring the continuity in position, velocity and acceleration on all the trajectory. The initial configuration is $[0.0 \;\; -1.57 \;\; 1.57 \;\; 0.0 \;\; 1.57 \;\; 0]$ $[rad]$ (fig. 12a).

Let us emphasize that the two parts of the trajectory have the same rotation vector $u = [0 \;\; 1 \;\; 0]_{|R_6}$, which is parallel to the z_5 axis. The angular velocity is therefore always in the reachable space. In addition, the evolution of the coordinates of the z axis of $R_d(t)$ guarantees that, during the reference trajectory tracking, z_4 and z_6 will be alined. So, the wrist singularity will be encountered and crossed 'transversally'.

Like in the equivalent case of the shoulder singularity (see section 5.1.2) the smallest value of λ ensuring the inversibility of $\frac{\partial e}{\partial q}$ is sufficient to obtain a smooth crossing. Thus, det_s and λ_{max} are fixed at 0.001 and T_λ at 100ms. The plots of figure 12b show the resulting evolution of the jacobian determinant, $det(J)$, and of $\lambda(t)$ during a simulation session. We can remark that the time at which the determinant vanishes is the same as the one where q_5 is equal to zero (fig. 12c), i.e z_4 and z_6 are aligned. Figure 12d shows the evolution of the x and z coordinates of O_6, which vary smoothly as expected.

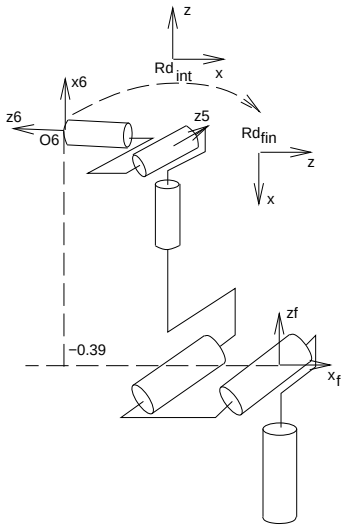
Trajectory 2: Let us now examine the case where the passage through a wrist singularity is accompanied by the demand of the realization of an angular velocity in the non attainable space. A reference trajectory satisfying these requirements is specified by

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & -1.0 & -0.39 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, \quad Rd_{int} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -0.26 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.53 \end{bmatrix},$$

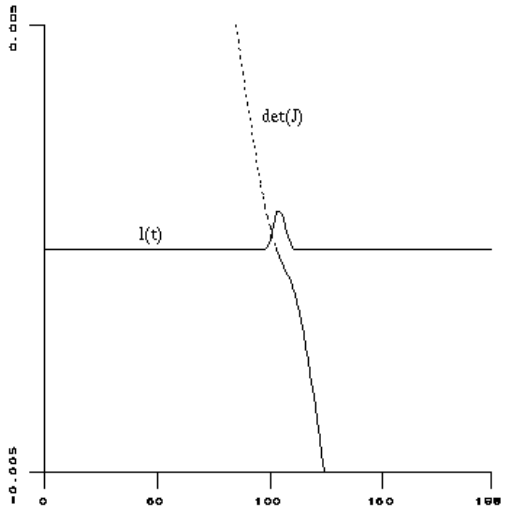
$$Rd_{fin} = \begin{bmatrix} 0.0 & -1.0 & 0.0 & -0.26 \\ 0.0 & 0.0 & 1.0 & 0.13 \\ -1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, \quad T = 3sec$$

and computed as the trajectory 1. In fact, while the first parts of these two trajectories are identical (fig. 14a), the angular velocity of the second part of the reference trajectory 2 has a non zero component in the non attainable space.

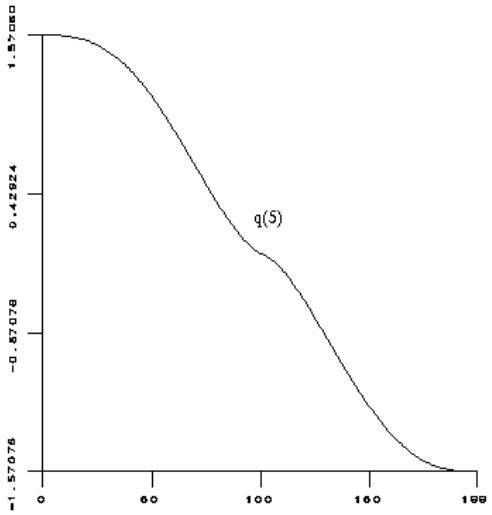
The simulation results, without singularity treatment, show clearly the effects of crossing the wrist singular points: the torque values applied on the fourth and sixth axis are excessive



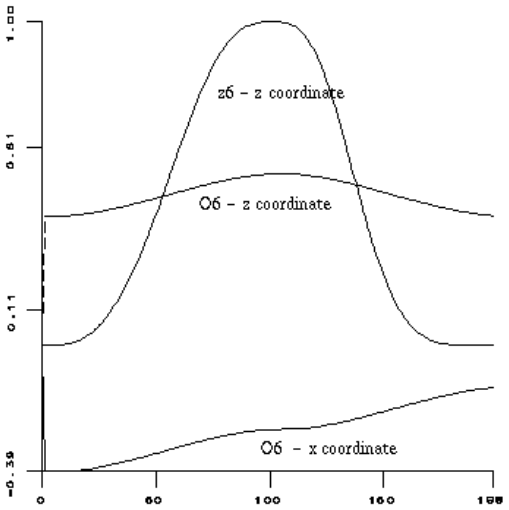
a) AID Initial Configuration



b) Determinant of the Robot Jacobian and $\lambda(t)$



c) q_5 evolution



d) x, z Coordinates of O_6 and z Coordinate of the z Axis of R_6

Figure 12: 'Transversally' Crossing Wrist Singularities with Singularity Processing (Trajectory 1)

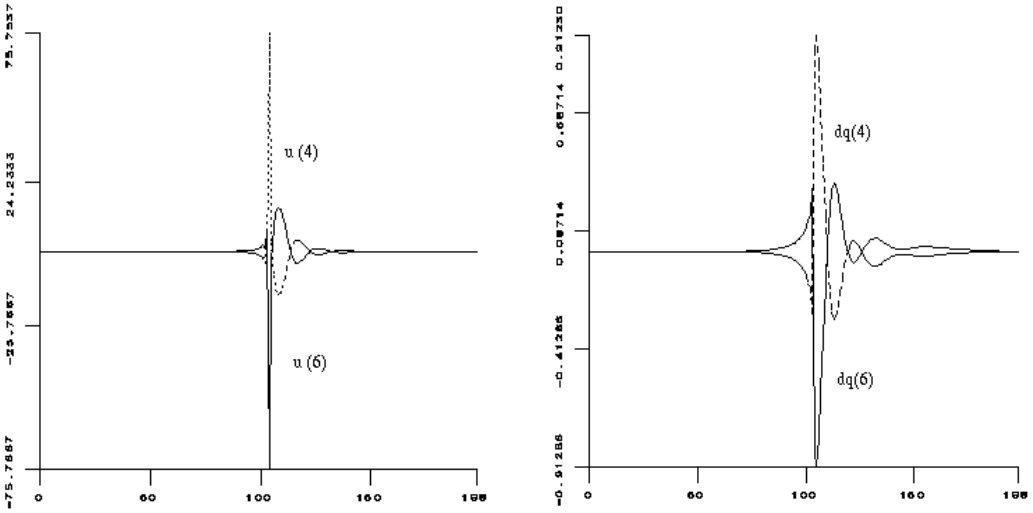


Figure 13: (a) Torques (b) Joint Velocities of Axes 4 and 6 Without Singularity Processing

(fig. 13) trying, as reported at the beginning of this section, to rotate quickly O_4 and O_6 in opposite directions.

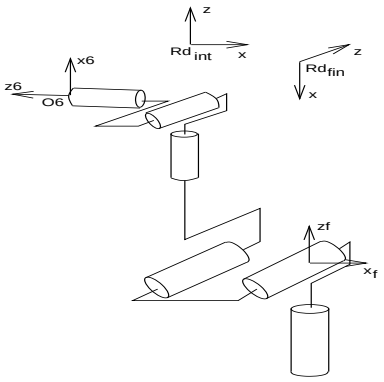
To obtain a smooth behaviour, det_s is fixed at 0.018 and λ_{max} equally at 0.009 with a T_λ of 700ms. The plots in figure 14b show the resulting evolution of the jacobian determinant $det(J)$ and of $\lambda(t)$ during a simulation session; the entrance in the singular region is detected at 0.9s. Now, the torque values stay in an acceptable range (fig. 14c and 14d), and the angular velocities are realizable. The plots in figures 14e and 14f illustrate the error in the tracking of O_6 and the evolution of the coordinates of the x axis of the R_6 frame, respectively.

5.2.3 Case Study 2: 'Staying On' the Singularity

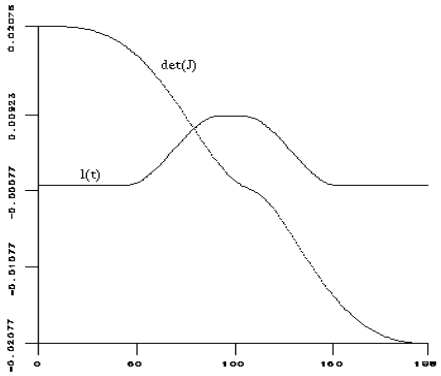
We now modify the second part of the trajectory 2 of the previous section in order to constraint the manipulator to stay on the wrist singular position, during a finite time interval, before leaving it. For this purpose we add a second intermediary point to the specification of the trajectory. The new reference trajectory is then defined by:

$$Rd_{init} = \begin{bmatrix} 0.0 & 0.0 & -1.0 & -0.39 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 0.0 & 0.0 & 0.40 \end{bmatrix}, \qquad Rd^1{}_{int} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & -0.26 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.53 \end{bmatrix},$$

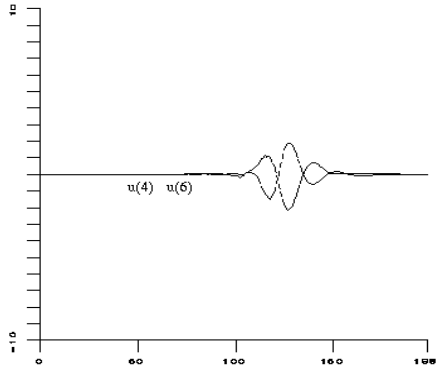
$$Rd^2{}_{int} = \begin{bmatrix} 0.973 & 0.0 & -0.227 & -0.373 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.227 & 0.0 & 0.973 & 0.457 \end{bmatrix}, \quad Rd_{fin} = \begin{bmatrix} 0.227 & -0.973 & 0.0 & -0.344 \\ 0.0 & 0.0 & 1.0 & 0.13 \\ -0.973 & -0.227 & 0.0 & 0.33 \end{bmatrix}$$



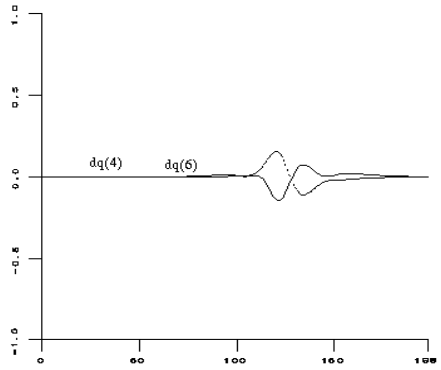
a) AID Initial Configuration



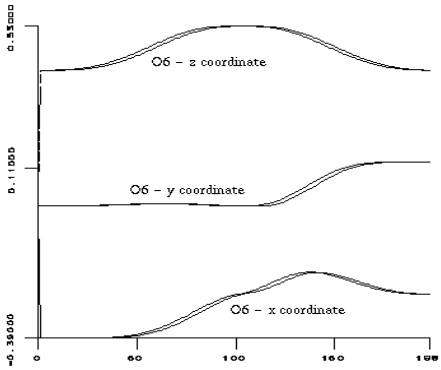
b) Determinant of the Robot Jacobian and $\lambda(t)$



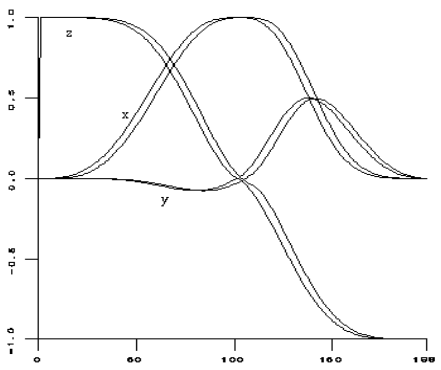
c) Torques on Axes 4 and 6



d) Joint Velocities of Axes 4 and 6



e) Current and Reference Coordinates of O_6



e) Coordinates of the x Axis of Frame R_6

Figure 14: 'Transversally' Crossing Wrist Sngularities with Processing (Trajectory 2)

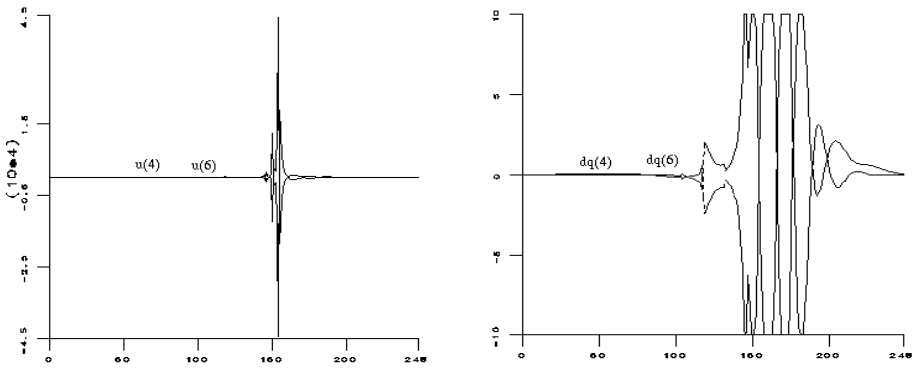
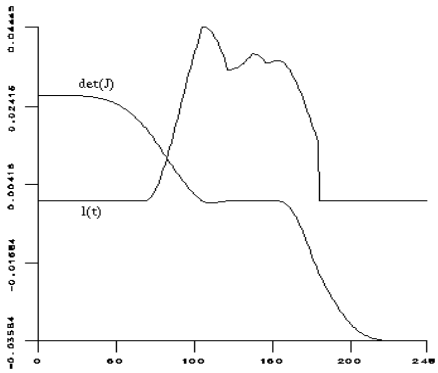


Figure 15: (a) Torques (b) Joint Velocities of axes 4 and 6 Without Treatment

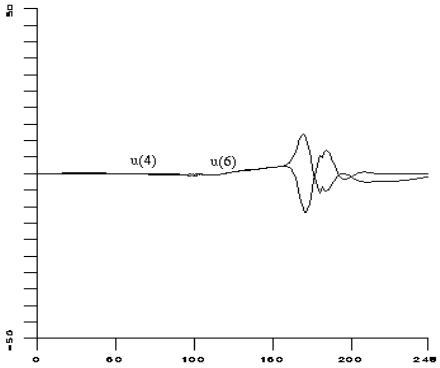
The three trajectory sections are interpolated with fourth degree splines for the first and final parts, and with third degree ones for the second, for ensuring continuity in position, velocity and acceleration on the intermediary points. The respective durations are of 1.5s for the first and last sections, and of 0.75s for the second one.

When the singularity treatment is not activated, the entrance in the singular region has the same effects on axis 4 and 6 as in the previous case (fig. 15).

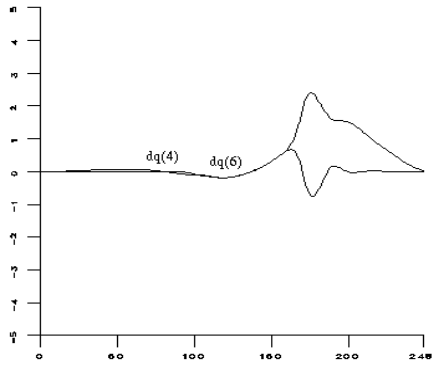
As usually, the same experiment was conducted with inclusion of the singularity processing. Parameter values are 0.019 for det_s, 0.05 for λ_{max} and 700ms for T_λ.The entrance in the singular region is detected at 1.05s. The induced evolution of $\lambda(t)$ is plotted in figure 16a. In addition, we can observe that, during the second part of the trajectory, the robot jacobian determinant is equal to zero, which means that the manipulator actually stays on the singular configuration. The control outputs varies as shown figure 16b and the joint velocities remain always feasible (fig. 16c). Figure 16a and 16b illustrate the tracking of O_6 and the evolution of the y and z coordinates of the z and y axis of the R_6 frame, respectively.



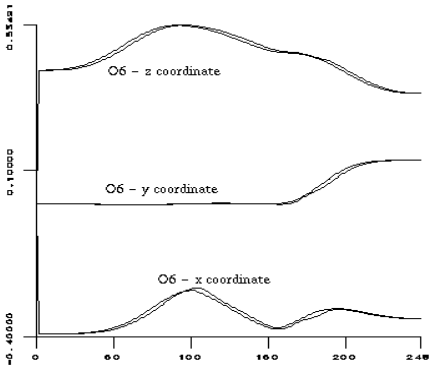
a) Determinant of the Robot Jacobian and $\lambda(t)$



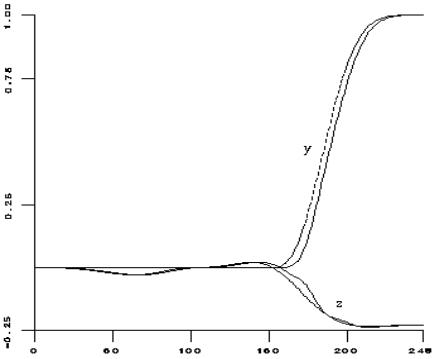
b) Torques on Axes 4 and 6



c) Joint Velocities of Axes 4 and 6



d) Evolution of O_6



e) Coordinates of the x axis of R_6

Figure 16: 'Staying On' Wrist Singularities with Processing (Trajectory 3)

5.3 Elbow Singularities

The elbow singularities occur when $q_3 = 0$ (fig. 1), which means that O_4 is on the boundary of its reachable space. These configurations can be identified by studying the position function of O_4, as described in section 5.1.1. However, we must notice that this kind of singularity is different in nature from the shoulder one: the set of the reachable points around a boundary point is not an open set.

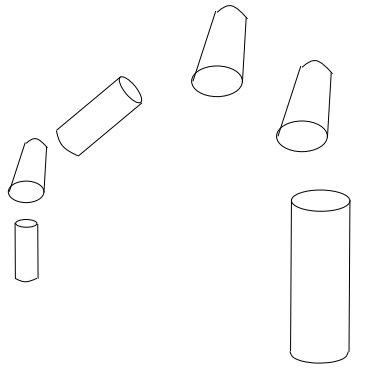
For studying the effect of the proposed method in the case of elbow singularities, we have defined a reference trajectory which drives O_4 to a boundary point of its reachable space and imposes to it a movement on the boundary; then, it leaves it and come back to the reachable space (fig.17a). It is specified by

$$Rd_{init} = \begin{bmatrix} -0.95 & 0.29 & -0.06 & -0.62 \\ 0.29 & 0.95 & 0.02 & 0.19 \\ 0.07 & 0.0 & -0.99 & -0.007 \end{bmatrix}, \quad Rd^1{}_{int} = \begin{bmatrix} -1.0 & 0.0 & 0.0 & -0.66 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 & -0.13 \end{bmatrix},$$

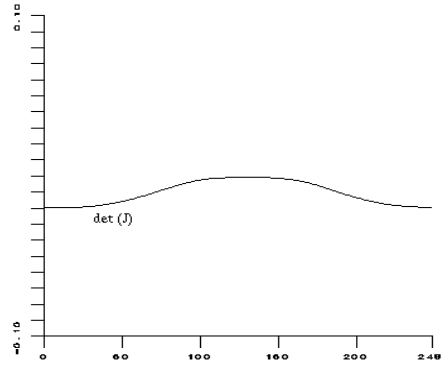
$$Rd^2{}_{int} = \begin{bmatrix} -0.99 & -0.09 & 0.0 & -0.65 \\ 0.09 & 0.99 & 0.0 & -0.065 \\ 0.0 & 0.0 & -1.0 & -0.13 \end{bmatrix}, \quad Rd_{fin} = \begin{bmatrix} -0.92 & -0.39 & 0.0 & -0.60 \\ -0.39 & 0.92 & 0.0 & -0.257 \\ 0.0 & 0.0 & -1.0 & -0.13 \end{bmatrix}$$

The trajectory computation guarantees a continuous profile; its total duration is of 3.75s, splitted in 1.5s for the first and last parts and 0.75s for the second one. The initial position is $\theta = [-0.3 \quad -1.2 \quad -0.3 \;\; 0.0 \;\; 1.5708 \;\; 0.0]$.

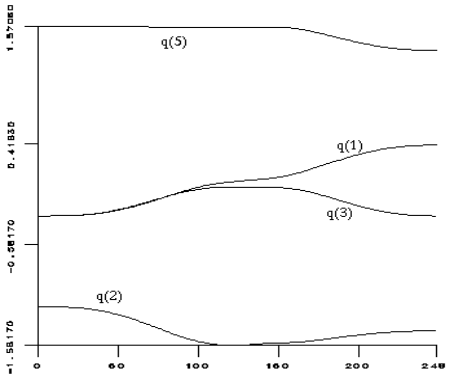
We can verify that during the second part of the trajectory, between 1.5s and 2.25s, $q(3) = 0$ (fig.17c) and $det(J) = 0$ (fig.17b) indicating that the corresponding configurations are singular. Besides, joint velocities varies smoothly and the reference trajectory is tracked as expected.



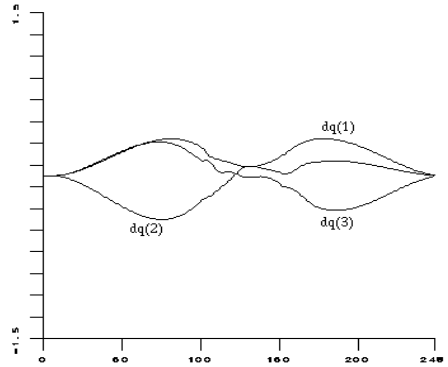
a) AID Initial Configuration



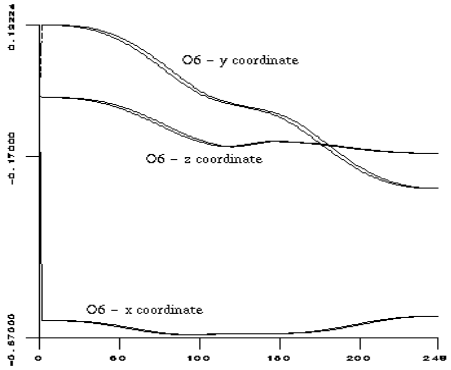
b) Determinant of the Robot Jacobian



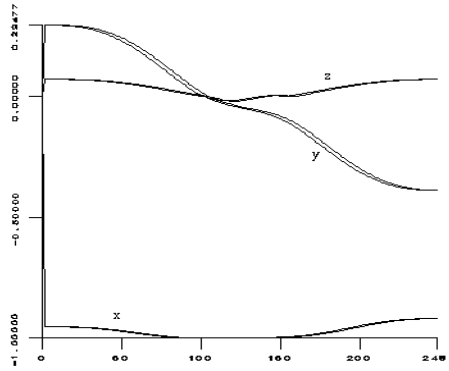
c) Joint Positions of Axes 1, 2, 3 and 5



d) Joint Velocities of Axes 1,2 and 3



e) Reference and Current Positions of O_6



f) Coordinates of the x Axis of R_6

Figure 17: Elbow Singularities with Processing

6 Conclusion

We have presented in this paper a general robot control method, inspired from [5] for coping with singularities during trajectory tracking. If we compare with other works reported in the literature, it can be seen that the proposed approach includes some specific contributions.

First, it should be emphasized that the method has a wide scope since it allows to consider any kind of singularities owing to the use of task functions. For example, the use of sensors, or the design of complex redundant tasks can be handled by the approach. A second point is that, even restricted to kinematic singularities, this work provides the reader with a clever justification of existing methods by setting the problem in a general way and making the possible approximations or the underlying assumptions explicit.

Another original issue in our sense concerns the implementation. In fact, it appeared to us that the ORCCAD framework proved very well-suited to the implementation of the proposed method. In particular, the concepts of type-1 exception, of observers and of robot-task automaton are particularly useful in that case. Moreover, the possibility of specifying synchronization, discretization and quantization parameters is clearly an help to the control designer, especially in such a case where conditioning and numerical aspects are dramatically relevant.

Finally, we should like to recall that we have tried to test the approach in the most significant cases, by selecting particularly difficult trajectories, not only transversal to the singularities like in general in the literature. We hope the results have proved the efficiency of the method, increasing in that way the robustness of the considered class of control methods.

References

[1] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling. *Numerical Recipes in C* Cambridge University Press,1988

[2] M. Le Borgne. *Quaternions et contrôle sur l'espace des rotations*, IRISA Report no 377, Rennes, France, Oct. 1987

[3] Y. Nakamura, H. Hanafusa. *Inverse Kinematic Solution with Singularity Robustness for Robot Manipulator Control*, ASME Journal of Dynamic Systems, Measurement and Control, vol 108, pp163-171, 1986

[4] C.W. Wampler, L.J. Leifer. *Applications of Damped Least-squares Methods to Resolved-rate and Resolved-acceleration Control of Manipulators*, ASME Journal of Dynamic Systems, Measurement and Control, vol 110, pp527-552, 1988

[5] C. Samson, M. Le Borgne, B. Espiau. *Robot Control: the Task Function Approach.* Clarendon Press, Oxford University Press, Oxford, UK, 1990

[6] C. Samson, M. Le Borgne, B. Espiau. *Robot Redundancy. An Automatic Control Approach.* NATO Advanced Research Workshop: Robots with Redundancy; Design, Sensing and Control, June 27-July 1, 1988, Salo, Italy

[7] S. Chiaverini, B. Siciliano,O. Egeland. *Controlling a 6-DOF Robot Manipulator in the Neighborhood of Kinematic Singularities*, 3rd Int. Symp. on Experimental Robotics, Oct 28-30 1993, Kyoto, Japan

[8] D. Simon, B. Espiau, E. Castillo, K. Kapellos *Computer-Aided Design of a Generic Robot Controller Handling Reactivity and Real-Time Control Issues*, IEEE Trans. on Control Systems Technology, vol 1, no 4, december 1993

[9] K. Kapellos, *Environnement de programmation des applications robotiques réactives*, PhD thesis, Ecole des Mines de Paris, France, nov. 7, 1994

[10] C. Astraudo, J.J. Borelly. *Simulation of Multiprocessor Robot Controllers*, Proceedings IEEE International Conference on Robotics and Automation, Nice, Mai 1992.

[11] F. Boussinot, R. de Simone. *The* ESTEREL *language*, Proccedings of the IEEE, 79(9):1293-1304, September 1991.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399