Contributions à la commande de robots sous contraintes

Sébastien Rubrecht

A dissertation submitted for the degree of

Doctor of Philosophy

of

Université Pierre et Marie Curie

in

Mechanics and Robotics

presented by

Sébastien RUBRECHT

CONTRIBUTIONS TO THE CONTROL OF CONSTRAINED ROBOTS

Defended on September 23^{rd}*, 2011*

Committee in charge

M. L. Baron	Professor at Ecole Polytechnique de Montréal	Referee
M. R. Boulic	Senior Scientist at Ecole Polytechnique Fédérale de Lausanne	Referee
M. T. Fraichard	Research scientist at Institut National de Recherche en Informatique et en Automatique	Examiner
M. G. Morel	Professor at Université Pierre & Marie Curie	Examiner
M. P. Bidaud	Professor at Université Pierre & Marie Curie	Adviser
M. V. Padois	Associate professor at Université Pierre & Marie Curie	Co-Adviser
M. M. de Broissia	Senior Engineer at Bouygues Travaux Publics Associate professor at Sherbrooke University Retired	Co-Adviser
M. Y. Measson	Business development of robotics and virtual reality interactive robotics unit, CEA LIST	Guest member

THÈSE de DOCTORAT

de

l'Université Pierre & Marie Curie

École Doctorale de Sciences Mécanique, Acoustique, Électronique et Robotique de Paris

Spécialité

Mécanique - Robotique

présentée par

Sébastien RUBRECHT

CONTRIBUTIONS À LA COMMANDE DE ROBOTS SOUS CONTRAINTES

Soutenue le 23 Septembre 2011

JURY

M. L. Baron	Professeur à l'École Polytechnique de Montréal	Rapporteur
M. R. Boulic	Maître d'enseignement et de Recherche à l'École Polytechnique Fédérale de Lausanne	Rapporteur
M. T. Fraichard	Chargé de Recherche à l'Institut National de Recherche en Informatique et en Automatique	Examinateur
M. G. Morel	Professeur à l'Université Pierre & Marie Curie	Examinateur
M. P. Bidaud	Professeur à l'Université Pierre & Marie Curie	Directeur
M. V. Padois	Maître de Conférences à l'Université Pierre & Marie Curie	Co-directeur
M. M. de Broissia	Ingénieur chez Bouygues Travaux Publics Maître de Conférences à l'Université de Sherbrooke A la retraite	Co-directeur
M. Y. Measson	Chargé d'affaires robotique et réalité virtuelle Laboratoire de robotique interactive, CEA LIST	Membre invité

Remerciements

La thèse est un moment particulier qui se termine par des remerciements. Cette occasion est rare, j'en profite.

Je tiens dans un premier temps à remercier mes encadrants, ils m'ont offert bien plus qu'un cadre professionel. Leur exigence, leur engagement et leur complémentarité ont composé le terreau de cette période épanouissante. Ils m'ont orienté, soutenu, encadré, accompagné, et ils resteront pour moi des exemples à plusieurs titres.

Un grand merci à tous ceux que j'ai croisé à l'ISIR, pour éviter d'en oublier il y a un nom, Trouville. J'aimerais adresser une mention particulière à ceux avec qui j'ai pu travailler, Juan, Puco, Ekta et Antoine. De manière plus informelle, Sébastien, Camille, Joseph et Bertrand.

Merci à Max, Olivier et Yvan du CEA-LIST qui m'ont laissé la possibilité d'éventrer les murs du CEA en renversant la maquette de tunnelier avec le robot Maestro (je maintiens que cela aurait pu arriver). Mes excuses à Xavier Lamy pour avoir cassé sa règle en plastique (je maintiens que c'est le robot pour le coup).

Merci à tous mes collègues chez Bouygues Travaux Publics, et en particulier le CCT pour leur accueil malgré mon "à bientôt" en réponse à leurs "à demain". Merci à Thibaut pour son aide spontanée.

Merci à tous mes amis, Gadz, Bretons, Nordistes, Lorrains, Bourguignons, Provençaux, Midi-Pyrénéens....

Merci à Cernay, même si ou parce que je n'ai jamais réussi à conserver leur attention en leur parlant de cette thèse, Franck, Vincent, Marion, Pilou, Denis.

Merci à ma famille, à cette promesse de toujours. Merci à toi Aurélie, cette thèse t'est dédiée.

Tout son corps remuait. De dos, on le voyait ainsi agiter bras et jambes, frapper des poings, frapper des pieds, à droite, à gauche, allongé pour posséder tout le clavier, et pesant du talon, parfois, de toutes ses forces, pour ébranler les bourdons. C'était un spectacle étonnant que celui de cet homme se dépensant ainsi, se démenant des quatres membres, pour mettre en branle la plus formidable machine sonore que l'homme ait imaginée.

– Maxence Van Der Meersch, *Maria fille de Flandre*

Abstract

To exploit the assets of robotics systems, missions are generally expressed through simultaneous or sequential goals to reach (objectives) while satisfying conditions on which no compromise is acceptable (constraints). These operational inputs being usually considered as unknown *a priori* and unpredictable, reactive control laws algorithms are used; these laws commonly satisfy constraints and objectives at their respective (potentially strict) priority levels, handle specific cases, manage transitions, etc. However, all these achievements focus on the *resolution* of the control problem, assumed to be *feasible*. The simple case of a joint submitted to both deceleration and position limits shows that constraints incompatibilities frequently occur.

To overcome such issues, a methodology is proposed to analyze the formulation of control constraints. In this thesis, we show that to ensure safety at the control level, either the permanent constraints compatibility should be proved, or the availability of alternative safe behaviors should be guaranteed. Some case studies involving common robotics constraints are proposed and lead to two main results: 1/ to remain compatible with joint accelerations limits, the intuitive expression of the joint position limits is modified; 2/ the operational acceleration being dependent of the robot configuration, its compatibility with obstacle avoidance cannot be proved, so dedicated alternative safe behaviors are proposed. These results are illustrated though experiments on a 6-DOFs manipulator.

The control problem being adequately formulated, its resolution algorithm has various specifications: 1/enforce compliance with the considered constraints; 2/fit the context efficiency requirements (real time, simulation, etc.); 3/find the optimal solutions; 4/offer a satisfying general behavior. The compliance with constraints being a prerequisite, compromises between those specifications are proposed. First, the elaboration of the Constraints Compliant Control law (CCC) based on the passive avoidance principle shows robust, efficient and performing features. It is particularly adapted to the task based design of a manipulator morphology, for which a huge number of robots are evaluated through trajectory trackings. Then, the use of a virtual displaced configuration in the control problem resolution enables to obtain a compromise between efficiency and optimality (safety being ensured anyway) through the first safe single iteration resolution method.

Keywords: Motion Safety, Robotic Constraints, Constraints Compliant Control, Inverse Velocity Kinematics, Redundancy Resolution, Evolutionary Design.

This thesis was hosted by *Bouygues Travaux Publics* - 1, avenue Eugène Freyssinet 78061 Saint Quentin en Yvelines - and the *Institut de Systèmes Intelligents et de Robotique* dependent of the *Ecole Doctorale de Sciences Mecanique, Acoustique, Electronique et Robotique de Paris* - 4, place Jussieu 75005 Paris France.

This thesis was partially financed by the Agence Nationale de la Recherche et de la Technologie (ANRT), within the CIFRE program. This thesis was involved in the Telemach Project granted by Agence Nationale de la Recherche, PSIRob 2007 Program.

Contents

Introduction

Robots are generally considered as efficient tools with high capabilities in terms of velocity, precision, dexterity and strength. To exploit these skills, the specifications of robotic missions are usually expressed through a set of simultaneous or sequential goals (objectives) to reach, *e.g.* "the end-effector of the robot should be placed at location A with orientation B", while satisfying a set of conditions on which no compromise is acceptable (constraints), *e.g.* "the robot should not collide with the environment". For historical reasons, the responsibility of the constraints satisfaction is generally attributed to the developer/trajectory designer, or even the user in teleoperation for example. This way of dealing with constraints is not suitable and the dissemination of robotics and the rise of robots autonomy impose to consider a rigorous management of the robots constraints directly at the root of the control law.

Reactive control loops are used in most robots control architectures; they are in charge of converting operational inputs (user specified objectives and constraints) into joint inputs at each time step. Algorithms are built to satisfy constraints and objectives specified at various (potentially strict) priority levels, to handle specific cases, to manage transitions, etc. However, all these achievements address only the resolution of the control problem; the solvability of the problem and the impact of the retained solution on the future control problems is rarely studied and constraints incompatibilities recurrently occur in robotics, which inevitably implies constraints violations, thus compromising safety. For example, to avoid an obstacle with a physical robot, the deceleration capabilities must be taken into account in the constraints formulation to ensure their simultaneous satisfaction. Using a control law able to solve a feasible problem is only the second condition to ensure safety at the control level; the first one is that the problem formulation maintains its permanent feasibility.

Once the problem is adequately formulated, its resolution algorithm must satisfy various specifications, among which: 1/enforce compliance with compatible constraints; 2/offer a satisfying general behavior; 3/find the optimal solutions; 4/fit the context efficiency requirements (real time, simulation, etc.). The first one is a prerequisite to ensure safety at the control level. The second one is often hard to formulate in terms of operational objectives but gathers smoothness considerations, the absence of oscillations, etc. The third one concerns the operational objectives and is more commonly treated in the literature. In this context, analytic model inversion and convex optimization methods offer many techniques which can be combined to obtain safe and performing control laws. However, the existence of more and more sophisticated robots asks for compromises between the efficiency of the controller and the optimality of the solutions.

The solutions and methods proposed in this thesis have been obtained by addressing the control problem as a whole, from its formulation to its resolution.

This document is organized in four parts. The first chapter describes the PhD

framework and exposes the context and problems addressed by this work. The second chapter details the formulation of the control problem to maintain its feasibility. The third chapter describes the constraint compliant control laws developed to ensure a safe and satisfying behavior. The fourth chapter details the results obtained in simulation for the task based design of a serial manipulator and some experiments on a 6-DOFs teleoperated manipulator. Finally, the last chapter proposes a conclusion and some perspectives.

Chapter 1

Context and Problems

This chapter is dedicated to the description of the thesis context. The thesis was hosted by Bouygues Travaux Publics and the ISIR laboratory at Université Pierre et Marie Curie, within the framework of TELEMACH, an R&D project dedicated to the feasibility of teleoperated maintenance in Tunnel Boring Machines (TBMs). TBMs are widely used for the excavation of small (diameter 1 m) to large tunnels (diameter 15 m and more). In this context, the task based design of the morphology of a robotic manipulator as well as its real time teleoperation in a cluttered environment were led by the ISIR laboratory. These two apparently unrelated topics raised a common and open research problem: the safe control of constrained robots.

1.1 TELEMACH: immersed robots for the teleoperated maintenance of TBMs

TELEMACH (TELE-operated MAintenance for TBMs Cutter-Head) is a research project proposed by Bouygues Travaux Publics in 2007 in response to a call for project of the Agence Nationale de la Recherche (ANR - France). This project lasted 30 months, from February 2008 to September 2010.

1.1.1 Topic and Context

TELEMACH finds its origin in the field of shield tunneling (see Fig. 1.1). There is a clear need to dig deeper and longer tunnels in urban areas, leading to major risks. Current working conditions are such that:

- The cutter head must be inspected frequently, if possible daily;
- In hyperbaric conditions, the intervention requires creating an air bubble in order to reach the cutter head tools. The realization of this air bubble takes time, and during this operation the front face (separation between the soil and the TBM) is not balanced optimally: an air bubble has an homogenous pressure profile whereas the front has an hydrostatic pressure profile, which is appropriately balanced during the excavation as the excavation room is full of mud and materials, see Fig. 1.2;
- Maintenance operations are ensured by operators in harsh conditions: the tools replacement imposes the operators to work in a hyperbaric air bubble within a

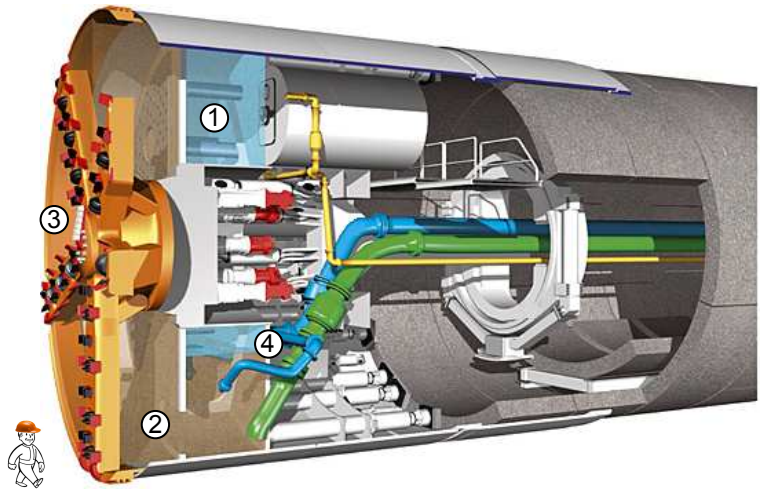

Figure 1.1: Slurry TBM during an excavation phase: an air bubble is maintained on the rear part of the excavation room to control the pressure of the mud. During the maintenance phase, the air bubble (1) is extended to the whole excavation room (2) to let the operators access the tools on the cutter head (3). The pressure bulkhead (4) separates the hyperbaric area and the rear of the TBM at atmospheric pressure. The diameters retained for the project are 9 m and larger.

narrow and dirty area only reachable by airlocks. The operators manipulate heavy tools (for example a disc cutter weighs 150 kg and is roughly an iron cube of 60 cm^3) in a confined area.

- Hyperbaric operations become very complex, long and expensive for pressures superior to 3 bars, which is common (equivalent to 30 m under the earth level). Up to this pressure, operators can be trained to intervene in the excavation room; nonetheless they must respect decompression cycles. Above 3 bars, it is mandatory to resort to professional divers (Le Péchon *et al.* 2000). Above 4 bars, exceptional measures of deep diving have to be taken: life in a hyperbaric caisson and displacement in a hyperbaric shuttle, helium based breathable gas ... As an example, on the Westershelde site (Holland), teams of 9 divers remained 3 weeks under a pressure of 6.5 bars (see Fig. 1.2).

TELEMACH is a feasibility study for replacing human interventions in the excavation room of TBMs by teleoperated maintenance. The missions to carry out (inspection, cleaning, tools replacement) would be performed by dedicated robotic systems: articulated arms for inspection, tools changer, heavy loads carrier, conveyor and automated airlock doors. The operator would act remotely through teleoperation and would be assisted in real time by a mobile camera, a virtual environment and force feedback. One of the particular aspects of the project concerns the performance of operations immersed in the mud (bentonite): it makes the air bubble unnecessary, which decreases the major risk of collapsing of the front face while reducing the idling time.

Figure 1.2: **Left:** the front face needs a hydrostatic profile pressure to be adequately balanced: using an air bubble constitutes a risk as the front face is not stable. **Middle:** Mobile airlocks used by divers for depressurization phases. **Right:** Operator during a depressurization phase.

1.1.2 TELEMACH consortium

TELEMACH was a feasibility project gathering various partners (see Fig. 1.3). The project was led by Bouygues Travaux Publics (BYTP) – one of the leading contractors in tunnel construction. The TBM design aspects were ensured by Herrenknecht (HK) – the world leader in the design and manufacturing of TBMs. The architectural development studies were led by Cybernétix (CYX) – a leading SME specialized in remote operations for this industry. The Commissariat à l'Energie Atomique et aux Energies Alternatives, Laboratoire d'Intégration des Systèmes et des Technologies (CEA-LIST) was responsible for the definition and development of the "hand-eye" support functions for the operator. The Institut des Systèmes Intelligents et de Robotique (ISIR), a university based research laboratory brought its support for the initial design and control of the robotic manipulator.

Figure 1.3: Consortium of TELEMACH and main protagonists during a TBM visit on A41 site (Geneva).

1.1.3 TELEMACH achievements

TELEMACH ended in September 2010. The feasibility was demonstrated through various achievements:

- **The TBM was modified and an original heavy load handling system was developed.** A whole flow study (man/tools/equipment) led to a complete reorganization of the airlocks. Security constraints such as the capability to reach any area with a stretcher were taken into account. The equipment airlocks, as the whole logistic chain, were automated to avoid direct man interventions. All actuators were deported in the rear part of the TBM at atmospheric pressure. The Disc cutter Tool Changer (DTC), a parallel robot (Stewart platform), was developed to unmount/extract worn disc cutters and insert/mount new disc cutter (Fig. 1.4). A mobile platform enables to reach the disc cutter casings by moving along a vertical wall. The whole system is under patent application.

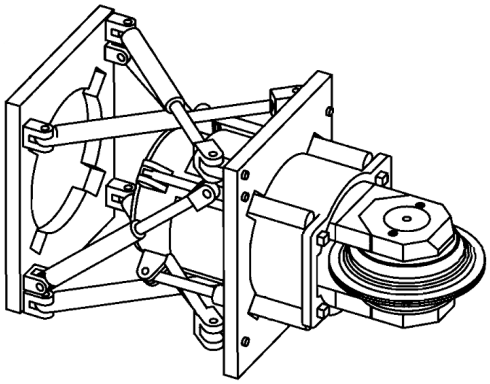

Figure 1.4: Disc cutter Tool Changer.

- **New tools for Interactive Teleoperation were developed.** A real time loop between a supervision system, the master arm and the slave arm was established with a satisfying transparency/stiffness compromise for the user. This framework enables to generate force feedback from a real time updated virtual environment. It opens the way to many applications developed along the project, for example:

  - Dynamic anti-collision (David *et al.* 2011): at each time step, a distance calculus between the robot and the environment is used to generate avoidance forces on the master arm proportional to the proximity to the environment.
  - Point of view optimization: a robot holding a camera holder is directly controlled by the scene tracking while avoiding the occultations thanks to an active anti-collision between the environment and a virtual cylinder representing the camera field of view.

  All the developments have been experimented on a full scale mock-up (see Fig. 1.5) with a Maestro[1] articulated arm (David *et al.* 2007).

[1] http://www.cybernetix.fr/Hydraulic-arms

Figure 1.5: Full scale mockup of a 1/4 TBM cutter head.

- **Intervention in the mud was assessed.** Intervention in the mud offers a major security advantage since pressure is balanced at the front (see Fig. 1.2). In this scope both touch-based recognition and ultrasound perception were assessed. On the one hand, peg-in-hole experiments ensured the precise localization of landmarks in the excavation room. On the other hand, the mud has been characterized (absorption, dispersion, sensitivity to pressure and temperature) and proper parameters were found to lead to satisfying observation of the tools.

Other TELEMACH contributions are part of this thesis work and are described in the next sections.

1.2 Control problems met in TELEMACH

Through TELEMACH, general control problems are addressed. The 2 main topics of TELEMACH dealing with control are:

- the task based design of a manipulator thanks to a genetic algorithm which fitness is a trajectory tracking;
- the real time control of this manipulator in a cluttered environment.

1.2.1 From a design to a control problem

The kinematic design of robotic manipulators is often seen as one among numerous applications of engineering design. However, the workspace of the excavation chamber is cluttered, so usual design techniques are inefficient and time-consuming. Moreover, the complexity of the TBM environment and the tasks to be accomplished induce a high number of potential design solutions among which the best ones may, given their originality with respect to usual design problems solutions, probably not arise using

classical design methods. The use of dedicated CAD tools may help to numerically discard some of the potential solutions, but checking each robot candidate with respect to a representative subset of tasks and environments still remains a complex and time consuming work.

Instead, it is proposed to follow an approach where the design process is considered as a multi-objective optimization problem: tasks and constraints are formulated in terms of functions to optimize and constraints to satisfy. Such a formulation allows the automation of the design process in the preliminary phase. Given a family of automatically obtained solutions, the so-called classical design methods can then be used to converge towards a practical solution.

The retained design process is an example of task-based design carried out thanks to an evolutionary process (*e.g.* Salle et al (2004)). Robot morphologies are generated thanks to a genetic algorithm which evaluation step (*fitness function*) aims to qualify the ability of a robot to carry out a maintenance mission in the TBM. So, a relevant trajectory has been defined in the simulation environment and the fitness function consists of a trajectory tracking. The objectives (*indicators*) retained in our problem are voluntarily simple and of a single dimension (no weighted sums representing *a priori* tradeoffs between different variables). The trajectory tracking quality but also intrinsic parameters, such as the number of DOFs, are evaluated.

Each robot morphology must be rated through a *fair* evaluation, *i.e.* an evaluation which exploits its physical characteristics optimally. In that scope, the control law used in the simulation is an essential element: it is responsible for the robot behavior, and consequently its scores. As a result, some expected properties are identified at the control level.

- **Genericity in robots**. Each morphology that can be described by the design process must be controllable by the control law without any specific restriction (geometry, degree of freedom, ...). The skills of each morphology must be exploited with equal chances.

  The genericity in robots demands a generic control law, the robot being *redundant* (the number of DOFs needed to obtain the desired motion is lower than the actual number of DOFs of the robot) or not. The terms of the control laws should be automatically computed without particular tuning;

- **Genericity in situations**. The control law must take into account and manage appropriately specific conditions such as singular configurations, constrained areas, oscillating behaviors, ...

  The genericity in situations requires the potential singularities to be automatically treated, and a correct behavior (no oscillations, smoothness) for all cases. If matrix singularities can be treated by the approximations of Damped Least Square Inversion (DLS, Wampler 1986 and Nakamura *et al.* 1986), the solution to a smooth and oscillations-free trajectory needs to be formalized and solved;

- **Representativity**. The control law used in the fitness should produce realistic behaviors to get meaningful evaluations and meaningful results. In particular, collisions should not occur in the simulations, as they cannot occur in reality.

  Representativity induces the need to handle constraints at the top priority level (the robot should not collide and, if possible, track the trajectory). Most of these

constraints are expressed through inequalities, which induces a non linearity in the problem. As a consequence, simple inversion operators are not able to satisfy those constraints and the resort to iterative algorithms is needed[2] to know which constraint can be taken as an equality (*active* constraint) and which cannot;

- **Coherency with indicators**. Robots are evaluated though performance indicators (for example trajectory tracking error, effort transmission capabilities, ...). The control law must take these indicators into account.

  Coherency with indicators induces the ability to deal with multiple objectives (minimization of the trajectory tracking error, getting close to a reference configuration, etc.). These objectives does not have the same priority, thus the controller should satisfy a strict priority between them. As for the constraints avoidance, getting closer to the reference configuration should be carried out only if it does not impact the trajectory tracking error minimization. However, the main difference between the constraints and the objectives is that the first one usually prevents motions while the second one generates motions. The satisfaction of multiple objectives in presence of constraints can be treated by a sequence of Quadratic Programs (QP), but it may turn out time consuming, which goes against **efficiency**;

- **Efficiency**. As a huge number of individuals are evaluated, the control law must not be time-consuming.

1.2.2 Reactive control for teleoperation

The working conditions in the excavation room of a TBM are not only harsh (cluttered area, dirty environment, hyperbaric pressure) but also critical in terms of safety. Actually, any failure of the robot implies exceptional measures (*e.g.* freezing the soil) to get it out of the working area.

Similarly to the previous section, this context imposes specifications at the control level.

- **Constraints Compliance**. The controller must be able to satisfy an arbitrary number of constraints whenever the problem is feasible. The word "constraints" used in this thesis always refer to safety and, as such, the top priority when considering multiple tasks. Differences between what the user sends at the operational level and the actual motion are considered as acceptable if they are justified by constraints compliance. Conversely, differences between the input sent by the controller to the actuators and the motion actually carried out (called *control losses*) should be limited to negligible errors.

  Most constraints are naturally expressed by inequalities (*e.g.* limits on joint positions, velocities, accelerations, etc.). Moreover, the number of constraints being potentially higher than the number of DOF, it is not possible to satisfy properly the constraints by taking each of them as an equality tasks. As a result, the controller should be able to deal with inequality constraints.

- **Robustness**. The errors between the input sent by the controller to the actuators and the motion actually carried out should remain negligible. In particular, the

[2] except for the inversion operator of Mansard *et al.*(2009a), which is able to deal with simple robots submitted to a few constraints but turns out time-consuming in more complex cases

models validity conditions should be checked, especially when using locally linear models such as Jacobians.

Robustness requires in particular a proper management of the model inversion singularities.

- **Multi-objective**. The working environment is cluttered and the missions to carry out in the excavation room are complex: cleaning, inspection, manipulation, etc. The robotic system used to achieve these missions is complex and thus able to carry out several objectives simultaneously. As a consequence, a multi-objective control structure is required.

- **Real-time**. Indeed, teleoperation requires real-time control laws, *i.e.* a computation time which order of magnitude is 1 ms.

As a complement to these specifications, a particular aspect of the problem must be considered. Actually, the control problem is often considered as a *problem to solve*, which implicitly relies on the assumption that "the problem is feasible" (which is concretised by the existence of control solutions). However, even when considering the reactive aspect of the control (for which a new control problem should be solved at each time step), it is fundamental to notice that the choice of motion at a given time step has an effect on the behavior of the robot and thus on the feasibility (or unfeasibility) of the incoming control problems. For example, accelerating the robot end-effector toward a wall is a constraint compliant motion until the very last time step before the collision. As a consequence, it seems that the control problem is also a *problem to formulate* to maintain its feasibility along the time steps.

1.2.3 Specifications

As a conclusion to the presentation of the problems tackled in this thesis, even if the proposed topics (task based design and teleoperation) are different, they present similar requirements in control. The resulting specifications are:

- **Feasibility awareness**: guaranteeing the relevance (representativity) of GA evaluations or ensuring the safety of a teleoperated mission requires solving the control problem while satisfying constraints. In order to maintain the feasibility of the problem, the controller should ensure that the control problem of the incoming time steps will be feasible;

- **Constraints compliance**: for the same reasons, the control law should be able to solve a constrained problem whenever it is feasible;

- **Multi-objective management**: the coherency between indicators and the achievement of complex teleoperation missions require a hierarchized multi-objective structure;

- **Robustness and genericity**: to ensure the same quality of control whatever the robot and the situation, the control law should be robust and generic;

- **Efficiency**: for a huge amount of simulations or for real time teleoperation, efficient control laws are expected.

1.3 State-of-the-art and contributions

This section exposes the state-of-the-art of the tackled domain and presents the contributions of the thesis.

1.3.1 Scope of the work

The topic considered in this thesis is usually referred as reactive Inverse Velocity Kinematics (IVK): at each time step, it is assumed that an operational velocity input $\dot{\boldsymbol{X}}_{des}$ (size m) is received by the controller, which is expected to send a joint velocity output $\dot{\boldsymbol{q}}_{des}$ (size n) to the actuators. $\dot{\boldsymbol{X}}_{des}$ belongs to the *operational space* while $\dot{\boldsymbol{q}}_{des}$ belongs to the *joint space*. The examples used along this document are supposed to have exclusively single-DOF actuated rotational joints, even if the proposed approaches are able to deal with multi-DOFs usual joints. The number of objectives (m) concatenated in $\dot{\boldsymbol{X}}_{des}$ can be greater, equal or inferior to the number (n) of Degrees Of Freedom (DOFs) of the considered system: from a control perspective, the system can be over-constrained, fully constrained or redundant.

To address the resulting problem, three main families of techniques are identified:

- The recent development of online planning algorithms (*e.g.* Kröger 2010) tends to show that frontiers are getting thinner between planning and control. In spite of the context difference, various approaches can now be foreseen to tackle these problems, and in particular the constraints compliance, in which planning techniques have a strong history (Brady 1982).

- The convex optimization techniques, which is a mathematical domain on its own. Even if there are some robotics contributions in that field, these algorithms are often used as off-the-shelf tools, mostly for their efficiency features in the control of complex systems such as humanoids.

- The control based on model inversion in the lineage of Liégeois (1977) is the widest branch of the reactive IVK problems. This is the background of this thesis.

The management of the operational tasks into the controller scope (appearance, transitions, removal) is not addressed in this document. The interested reader can refer to the work of Mansard *et al.* (2007), Keith *et al.* (2009), Padois *et al.* (2007) and Salini *et al.* (2011). Similarly, the enforcement of the desired joint output by the actuators is out of the scope of this study.

1.3.2 State-of-the-art with respect to the specifications of the problem

The expected control law features are inequally treated in the literature. The **efficiency** is rarely treated directly, it is rather a set of precautions and limitations on the possible algorithms. The **robustness and genericity** have a considerable amount of literature: the use of Jacobians as generic model representations are common and widely used in reactive control to deal with any kind of robots (Liégeois 1977, Baillieul 1985). Similarly, the robustness to model inversion singularities has an important associated literature and solutions such as the Damped Least Square pseudo-inversion method introduced by Wampler (1986) and Nakamura *et al.* (1986) are considered as sufficient to reach the expected specifications.

The **simultaneous management of multiple objectives** is a well known problem in reactive control. The most popular method to deal with a set of objectives is a hierarchical framework (Liégeois 1977): objectives are prioritized and low priority objectives are carried out only if they do not impact the achievement of top priority objectives. This approach was recently generalized with the notion of *stack of task* (Mansard *et al.* 2007). Based on this framework, many kinds of inversion-based control problem resolutions were proposed: potential fields and gradient projection methods (Khatib *et al.* (1986) and Maciejewski *et al.* (1985)), weighted inversion techniques (Chan *et al.* 1995, Huo *et al.* 2011), clamping (Baerlocher *et al.* 2004), etc. These resolution schemes manage constraints avoidances as equality tasks and, the number of constraints being potentially higher than the number of DOF, they mostly fail in highly constrained cases and cannot lead to **compliance with any constraints**. Moreover, they are subject to discontinuities, which requires dedicated developments: adaptative gains for weighting techniques (Chaumette *et al.* 2001), transitions (Padois *et al.* 2007), progressive clamping (Raunhardt *et al.* 2007), dedicated inversion operator (Mansard *et al.* 2009a). Induced by the management of constraints thanks to equalities, most[3] of these methods rely on *active avoidance techniques*, *i.e.* approaches where constraints avoidance requires motions. As an alternative to this approach, Faverjon and Tournassoud (1987) proposed an obstacle avoidance technique included in a Quadratic Programming (QP) control law structure. This method limits the velocities toward obstacles using inequalities (*passive* avoidance), which is more likely to avoid the collisions whatever the number of obstacles. An extension of this approach based on convex optimization to the case of multiple objectives was recently proposed by Kanoun *et al.* 2009.

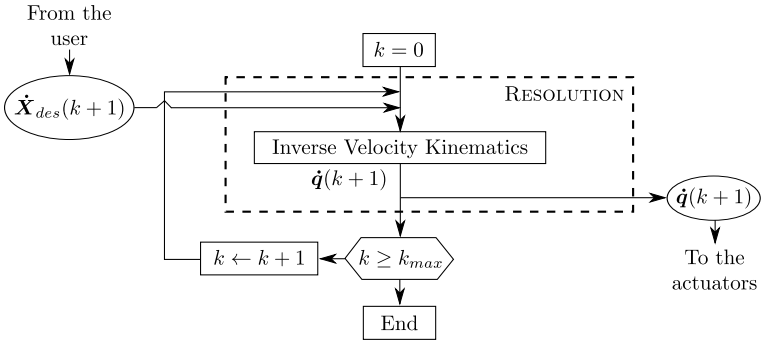

Figure 1.6: Reactive velocity kinematics control law algorithm. At each time step, the operational velocity inputs $\dot{\boldsymbol{X}}_{des}$ are converted into joint velocity outputs $\dot{\boldsymbol{q}}_{des}$.

Finally, **feasibility awareness** and, as an extension, safety at the control level, is a quite untackled field. Usual control approaches implicitly address the resolution of a control problem assumed to be feasible (see Fig. 1.6). In fact, most of the research work related to safety at the control level is led in the field of mobile robotics, *i.e.* single body mobile robots avoiding collisions: models are simpler, and the operational capabilities predictions are easier (operational deceleration limits do not depend on the robot configuration for example). As an example, the Dynamic Window Approach (DWA) (Fox *et al.* 1997) involves the acceleration limits of a mobile robot and ensures its safety in a fix environment. More recent developments in this domain are part of the framework based on the notion of Inevitable Collision State proposed by Fraichard *et al.* (2004).

[3]Excepted for weighting techniques and clamping. This point is detailed in chapter 3.

To the best of our knowledge, although this framework could be used to assess the safety of a wider scope of applications, 1/ it has never been applied to multi-body robots; 2/ it is limited to collisions avoidance with respect to dynamics, which can be formulated as the compatibility between the constraint of geometric collisions avoidance and acceleration limits. However, these are just two constraints among the many constraints that have to be faced in robotics: joint position, velocity, acceleration and torque limits (joint space), collisions with obstacles and forbidden regions (Cartesian space), contacts conservation constraints (Park *et al.* 2008), comanipulation and cooperation (Khatib *et al.* 2001), actuators temperature limits (Guilbert *et al.* 2008), etc. We can conclude that there is still a lack regarding robots safety (in particular for multi-body robots) when considering a large variety of constraints.

1.3.3 Contributions

The work carried out along this thesis addresses the control problem as a whole (see Fig. 1.7), from its formulation (chapter 2) to its resolution (chapter 3). Finally, chapter 4 exposes the results through simulations and experiments.

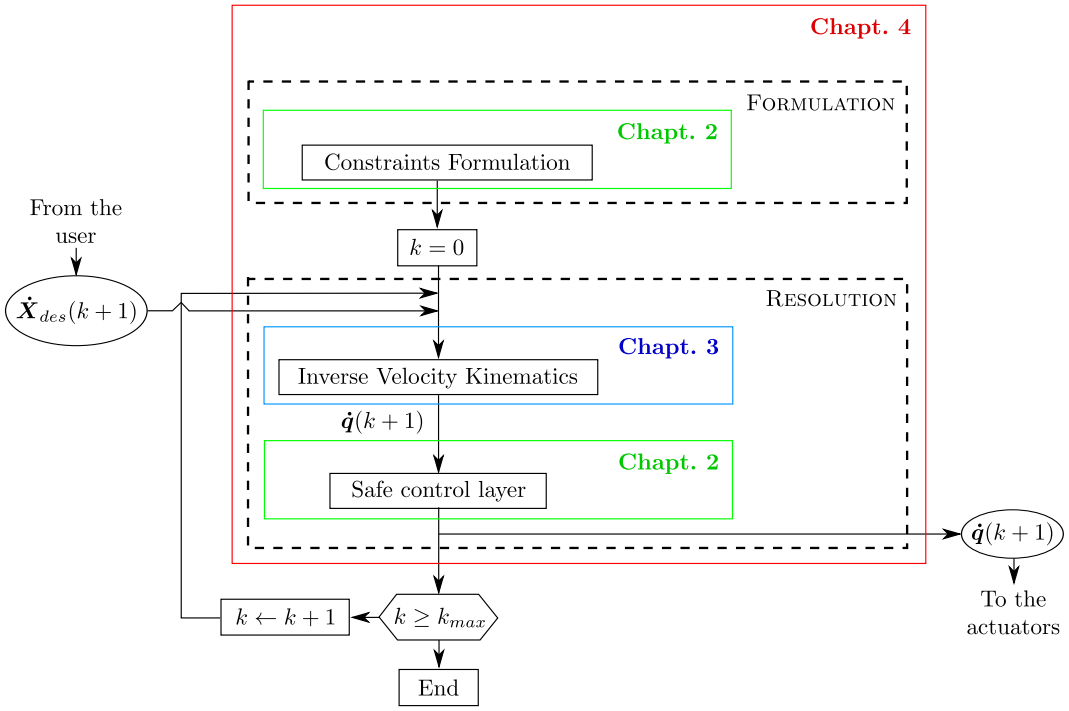

Figure 1.7: Organization of the manuscript. Chapter 2 deals with the formulation of the control problem and the safety aspects. Chapter 3 focuses on the resolution of the control problem. Chapter 4 validates the work exposed on the previous chapters through simulations and experiments.

The aspects of safety and the formulation of the control problem are tackled in chapter 2. In this chapter we propose a formal approach to ensure safe behaviors of multi-body robots in a reactive control framework, thus addressing **feasibility awarness**. This approach focuses on the constraints expression; the compatibility between these constraints is studied, and safe alternatives are ensured when compatibility can-

not be established. A complete case study involving obstacles, joint position, velocity and acceleration limits illustrates the approach. A particular method is developed to take full advantage of the usual avoidance techniques while maintaining safety. The corresponding work was recently proposed for publication in *Autonomous Robots* and is under second review[4] (Rubrecht *et al.* 2011).

The aspects of constraints compliance and the resolution of the control problem are tackled in chapter 3. First, the Constraints Compliant Control approach is introduced. It relies on a passive avoidance scheme (no motion generation for constraints avoidance) on a limited number of constraints selected from a vicinity analysis. A scaling solution, based on the feasible motions with respect to the constraints, enables to reach the frontiers of the workspace. Second, the displaced configuration method extends the Constraint Compliant Control law to any set of constraints and proposes a compromise between optimality and efficiency. These contributions have respectively been published in the ARK'2010 (Rubrecht *et al.* 2010a) and IROS'2010 (Rubrecht *et al.* 2010b) international conferences.

The last chapter is dedicated to the results. First, simulations validates the approach proposed to solve the control problem in a satisfying manner. Two missions described as sequences of keyframes are simulated to compare the performances of the proposed control law to state-of-the-art control laws. The obtained computation times remain acceptable to consider a use in real time. Then, the results obtained in task-based design are proposed. As the solution space is likely to be shaped strangely due to the particular working area, a special attention is paid to support the evolutionary algorithm exploration and avoid negative impacts from the problem formulation, the fitness function or the evaluation. In that respect, a specific genome able to encompass all cases is set up and a constraint compliant control law is used to avoid arbitrary robots penalization. The presented results illustrate the methodology adopted to work with the developped evolutionary-aided design tool. Finally, experiments involving a 6-DOFs manipulator operating in front of a TBM cutter head mock up confirms the reliability of the approach to safety and validates the expected performances. The work on design was published as a book chapter (Rubrecht *et al.* 2009) in a volume of the Springer Studies in Computational Sciences dedicated to evolutionary robotics.

[4] "accepted pending revision"

Chapter 2

Motion safety and constraints expression for multi-body robots

A usual robotic motion chain involves 3 main actors: the *user*, which sends operational inputs to the robot, the *controller*, which transposes these inputs into the joint space, and the *physical system* which carries out the motion (see Fig. 2.1). The principle of safety applied to the control field requires a degree of autonomy of the controllers. Given a user specified operational input, the controller is in charge of computing the joint input so that it carries out the desired motion *at best*. Obviously, the *best* motion tracking is not only the closest to the one specified by the user, but above all a *controlled* one, *i.e.* a motion for which the desired joint input sent by the controller (motion intended by the controller) is actually carried out by the physical system. Presently, it may not be the case for example if the controller does not integrate the capabilities of the physical system. In that perspective, a relevant and commonly used approach consists in limiting the exploitation of the system model to a domain on which it is reliable (usually referred as the *validity* domain of the model). However, even over this domain, the permanent satisfaction of the intuitive expression of usual constraints (*e.g.* $\boldsymbol{q}_m \leq \boldsymbol{q} \leq \boldsymbol{q}_M$, $\dot{\boldsymbol{q}}_m \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_M$, $\ddot{\boldsymbol{q}}_m \leq \ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{q}}_M$) turns out to be insufficient to prevent their violation (see Fig. 2.2).

Consequently, it is deemed that the control approach should be composed of 2 steps. At each time step, the controller should:

1. Find the domain in the control variables space for which safety and constraints are satisfied: the **control problem formulation**;

2. Find a control vector solution within this domain: the **control problem resolution**.

Chapter 2 deals with the control problem formulation and chapter 3 with the control problem resolution. As an illustration, if the control problem is formulated as a Quadratic Program (QP), the aim is to define the equalities/inequalities subject to which the problem is to be solved. Once this is done, whatever the resolution method, if it is able to solve a *feasible* problem (in which constraints are compatible), then safety is ensured. Chapter 3 is dedicated to the control problem resolution assuming that the expression of constraints is compatible and thus that safety has been ensured beforehand; in the case of a QP, chapter 3 would be focused on the optimization algorithm itself.

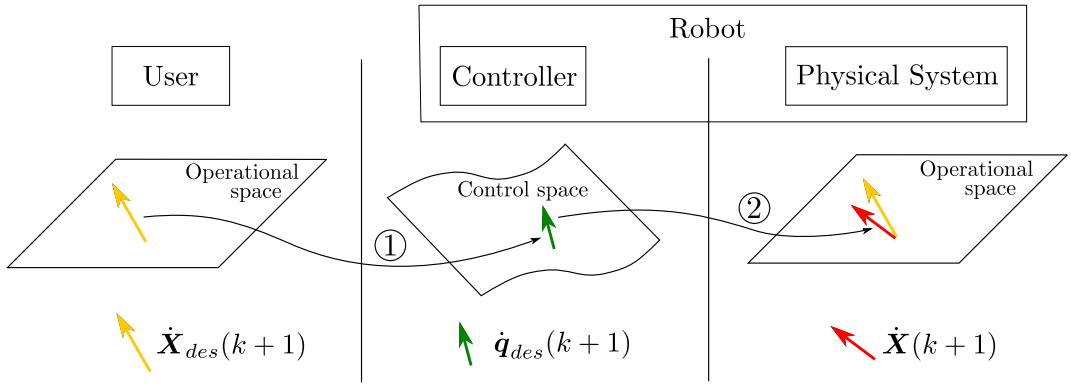

Figure 2.1: Motion chain: at each time step, the user or a high level loop specifies an operational desired value for the next time step in the operational space (in this case, desired velocity in the Cartesian space $\dot{\boldsymbol{X}}_{des}(k+1)$). This desired input is translated into the joint space by the controller (1), yielding a joint velocity vector $\dot{\boldsymbol{q}}_{des}(k+1)$. Finally, the motion is carried out (2) by the physical system, yielding a motion which may be different from the one expected by the user.

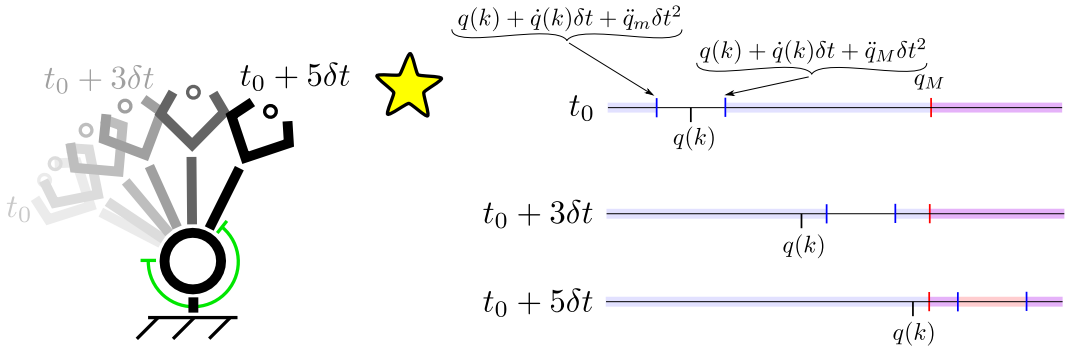

Figure 2.2: Example of incompatibility between constraints for a 1-Degree of Freedom robot trying to reach the star-shaped object. The right part of the Figure is the sequence of motion and the left part is the representation of the system configuration at times t_0, $t_0 + 3\delta t$ and $t_0 + 5\delta t$ in the joint space. At time t_0, the robot velocity is null, and the next configuration can be chosen in the neighborhood of the current configuration $q(k)$ within the interval let free by the constraints of joint position and acceleration limits. The terms $q(k) + \dot{q}(k)\delta t + \ddot{q}_m(k)\delta t^2$ and $q(k) + \dot{q}(k)\delta t + \ddot{q}_M(k)\delta t^2$ are respectively the configurations induced by a full acceleration and full deceleration (approximation by finite differences). At time $t_0 + 3\delta t$, the neighborhood of the configuration $q(k)$ is no longer reachable as it does not comply with the joint deceleration limit. Finally, at time $t_0 + 5\delta t$, there is no more constraint compliant solutions (no free interval in the joint space) and the collision with the joint position limit cannot be avoided; this constraint incompatibility should have been foreseen before the whole motion generation.

The first section of this chapter discusses the notion of safety at the control level. Secondly, the concept of *extended state* (e-state) is introduced and a definition of safety at the control level is proposed. The third section is dedicated to the methodology proposed to guarantee safety. It is shown that safety can be ensured either by making the constraints compatible with each other or by guaranteeing the permanent possibility

to resort to an alternative safe behavior.

2.1 Safety at the control level

This section is focused on the notion of safety at the control level. In a first part, general safety criteria are cited. Then, a literature review highlights the present lacks in this particular domain. In a third part, safety is discussed and a safe motion chain structure is proposed. Finally, an outcome sums up the objectives and the assumptions of the study.

2.1.1 Criteria for safety

The notion of *safety* for a system is a principle applied at various levels. At the design level, safety is often integrated directly in the system (Ikuta *et al.* 2003, Zinn *et al.* 2004, Haddadin *et al.* 2010). At the control level, the work related to offline optimal trajectory planning is closely linked to joint constraints management (Brady 1982, Biagiotti *et al.* 2008) but, despite the context differences[1], its recent adaptations to online frameworks (Kröger 2010) exhibits some similarities with reactive control techniques. In a strictly reactive context, safety has been neglected for a long time. Recently, Fraichard (2007) proposed 3 criteria to ensure safety:

1. To decide its future motion, a robotic system should consider its own dynamics;
2. To decide its future motion, a robotic system should consider the future behavior of the environment;
3. To decide its future motion, a robotic system should reason over an infinite time-horizon;

These criteria are formulated at a high level and seem to express rather well the common feelings safety refers to. In his work, Fraichard assesses single body mobile robotics control approaches with respect to his definition of safety.

2.1.2 Common control approaches for collisions avoidance

The following literature review evaluates multi-body robots control laws with respect to Fraichard's criteria. It does not detail the way the control problem is solved (for this, refer to chapter 3) but rather focuses on their adequacy with the safety criteria.

To illustrate the topic, let us consider a robot moving close to one of its joint position limits (as in Fig. 2.2). To handle safety, the controller must consider its ability to avoid the joint limit (and its implication on the joint acceleration or torque limits) as a prerequisite to motion. However, to our knowledge, the avoidance strategies in the literature do not involve this type of information. As a result, these strategies are either unsafe, and consequently cause joint position limits violations, or too cautious and then waste the system workspace, which may prevent optimal mission performance in cluttered environments for example.

Safety in itself is rarely addressed directly in the literature; it is usually treated through collision avoidance or joint position limit avoidance for example. All the control

[1] most of these approaches are exclusively concerned with joint physical limits as operational constraints are managed by path planning

approaches from which we may expect safety rely on the model of the system, which can be taken as the velocity kinematic model for example

$$\dot{\boldsymbol{X}}_{des} = \frac{\partial \boldsymbol{X}_{des}}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} = J_O(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{2.1}$$

where $\dot{\boldsymbol{X}}_{des}$ is the operational desired velocity, $\boldsymbol{q}$ is the joint space configuration (referred later simply as *configuration*), J_O is the Jacobian associated to the operational objective and $\dot{\boldsymbol{q}}$ is the joint space velocity of the system.

2.1.2.1 Control based on model inversion

Most control laws in the literature solve the control problem by a direct inversion of the model. In the case of serial manipulators, the main families of avoidance technique are issued from Khatib *et al.* (1986) and Maciejewski *et al.* (1985), where forces or velocities based on the distances to obstacles are applied to the robot to generate an avoidance motion (active avoidance). All these approaches fail to ensure safety, even when not considering a moving environment. Two main reasons are responsible for this situation:

1. **Their framework is not adapted to respect an arbitrary number of strict constraints**. For all these approaches, the way to impose conditions on the system is done through equalities

$$\dot{\boldsymbol{X}}_{des1} = J_{O1}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{2.2}$$
$$\dot{\boldsymbol{X}}_{des2} = J_{O2}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{2.3}$$
$$\vdots \tag{2.4}$$
$$\dot{\boldsymbol{X}}_{desp} = J_{Op}(\boldsymbol{q}) \dot{\boldsymbol{q}}. \tag{2.5}$$

   If the number of equalities imposed on the system is inferior or equal to the system number of DOFs (n), the conditions are always (excepted for singularities) fully satisfied; if the number of equalities imposed is superior to the system DOF, the retained velocity vector $\dot{\boldsymbol{q}}$ minimizes the error, according to a predefined norm, to the desired motions[2], but are not able to satisfy all the conditions. As the robot dynamics involve at least $6n$ limits when considering position, velocity and acceleration limits, it is not possible to rely on such an approach to take into account the robot dynamics (criterion 1).

2. **These approaches do not consider acceleration or torque limits**. These approaches do not consider the robot dynamics (criterion 1). Most of them consider only the distance to the obstacle in the magnitude computation of the avoidance term. A step toward safety has been made by Choi *et al.* (2000) with the introduction of *collidability*, which involves not only distances but also velocities toward obstacles to have a better appreciation of the hazard of the situation. However, this information refines the avoidance strategy, but is not sufficient to guarantee a safe avoidance. For instance, what would happen if the operational deceleration capability suddenly decreased near an obstacle?

[2] weighted inverse techniques (Chan *et al.* 1995, Park *et al.* 2001, Huo *et al.* 2011) define the norm according to which to minimize the solution based on equalities to check.

2.1.2.2 Optimization-based control

To impose the satisfaction of constraints without imposing equalities to the system, and to profit from the fact that constraints are generally expressed by inequalities ($\boldsymbol{a}_{min} \leq \boldsymbol{a} \leq \boldsymbol{a}_{max}$), Faverjon and Tournassoud (1987) proposed an avoidance technique included in a control law expressed as a Quadratic Programming (QP)

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des} - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)|| \\ &\text{subject to} \qquad J_C\dot{\boldsymbol{q}}(k+1) - \boldsymbol{b} \leq 0. \end{aligned} \tag{2.6}$$

$\dot{\boldsymbol{q}}(k+1)$ is the velocity vector chosen for the next time step (control input vector), J_C is the Jacobian of constraints and $\boldsymbol{b}$ is the constraints limit vector. This method limits the velocities toward obstacles by inequalities (passive avoidance). As a result, QP techniques do not suffer from the first drawback of control based on model inversion; QP algorithms are now widely used in manipulators or humanoids controllers (Decre *et al.* 2009, Kanehiro *et al.* 2008, Salini et al. 2011). However, the avoidance methods still do not include dynamics (criterion 1): as for control based on model inversion, the joint torque or acceleration limits are not taken into account in the inequalities expressions. Consequently, compatibility between constraints (*i.e.* the existence of $\dot{\boldsymbol{q}}(k+1)$ such that $J_C\dot{\boldsymbol{q}}(k+1) - \boldsymbol{b} \leq 0$) is not guaranteed. The typical incompatibility case is the joint position limit violation due to limited accelerations illustrated by Fig. 2.2. If a joint gets close to one of its position limits with a high velocity, its deceleration capabilities may not be sufficient to avoid the collision with the position limit. As an example, a maximum deceleration 2 rad/s^2 imposed on a joint moving at 1.0 rad/s requires 0.5 s to actually stop; the distance travelled is 0.25 rad. This example illustrates the fact that satisfying at each time the joint position and the joint acceleration limits does not prevent from a constraint violation due to incompatibility. Usually, virtual envelopes are set up around the physical limits to absorb such violations. These envelopes do not guarantee safety and often artificially limit the performances of the robot. Relying on a safe approach taking dynamics into account would enable to reduce significantly those envelopes.

2.1.2.3 Main difficulty: operational capabilities

Actually, the main difficulty to consider operational deceleration capabilities in the control law is their dependency to the configuration of the system; thus, at a given time, the design of a safe deceleration scheme in a cluttered environment is an open problem. It results that, to our knowledge, **no control law for multibody robot passes criterion 1**.

In mobile robotics, the Dynamic Window Approach (DWA) (Fox *et al.* 1997) involves dynamics (by taking acceleration limits into account) and ensures safety in a static environment. The DWA is an example of method maintaining the compatibility between collisions avoidance and acceleration limits constraints. However, to our knowledge, 1/ this approach has never been applied to multi-body robots; 2/ this approach is limited to collision avoidance and joint acceleration limits. Indeed, these are just two constraints among the many constraints that have to be faced in robotics: joint position, velocity, acceleration and torque limits (joint space), collisions with obstacles and forbidden regions (Cartesian space), contact persistence constraints (Park *et al.* 2008),

comanipulation and cooperation (Khatib *et al.* 2001), actuators temperature limits (Guilbert *et al.* 2008), etc. We can conclude that **there is still a lack regarding robots safety (in particular for multi-body robots) when considering a large variety of constraints**.

2.1.3 Discussion of Fraichard's criteria

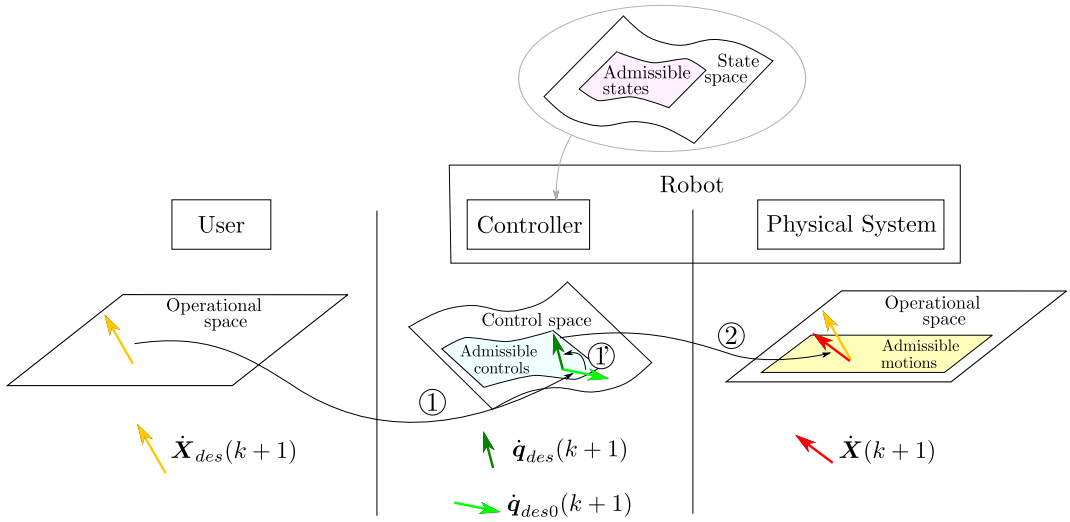

Figure 2.3: Proposed structure for a safe motion chain: at each time step, the user or a high level loop specifies an operational desired value for the next time step in the operational space (in this case, desired velocity in the cartesian space $\dot{\boldsymbol{X}}_{des}(k+1)$). This desired input is translated into the joint space by the controller (1), yielding a joint velocity vector $\dot{\boldsymbol{q}}_{des0}(k+1)$. As the controller knows the dynamics of the robot and the environment, it is able to determine the admissible state space (subspace of the state space) and thus the admissible control space (subspace of the control space) over which the velocity vector $\dot{\boldsymbol{q}}_{des}(k+1)$ must be chosen (1'). Finally, the motion is carried out (2) by the physical system, yielding a motion different from the one expected by the user, but known by the controller to be a safe motion.

Let us discuss the safety criteria proposed by Fraichard:

- To decide its future motion, a robotic system should consider its own dynamics;

First, the decision of a robotic motion should fall to the controller: through the operational input, the user indicates its specifications; however, among all the potential control vector solutions, this is the controller which possesses the information to assess what is the best solution and how to act accordingly (Fig. 2.3).

Then, one of the elements to evaluate a potential control solution is the robot's own dynamics, *i.e.* all the implications of the currently chosen motion on the future states of the robot. In this scope, the key elements are:

1. A proper perception of the system state (by sensors and state observers for example);

2. A representative model of the system (*i.e.* given a joint input, an acceptable matching between the predicted operational behavior and the real operational behavior for a wide domain of the state space called *nominal range*);

3. The consideration of all the limits imposed on the system state: the actuators features, the adequacy between the model and the real system, the limits imposed directly by the user (collisions, forbidden areas, velocity limitations), the missions specificities (*e.g.* cooperation missions), etc. Despite their differences, all these limits should be considered equally at the control level. A robot holding a cup of molten metal must not collide, spill it or reach forbidden velocities/accelerations to avoid spatters. Equivalently, a magnitude saturation is a model limit (control loss) and must be avoided.

- To decide its future motion, a robotic system should consider the future behavior of the environment;

Similarly to the robot, the future behavior of the environment should be estimated accurately (proper perception, representative model) to be treated as an input for the controller.

- To decide its future motion, a robotic system should reason over an infinite time-horizon;

As a consequence of integrating the dynamics in the decision making process, the time-horizon to be considered should be infinite. An instant-safe situation (instantaneous constraints satisfaction when not considering any time-horizon) is obviously not sufficient to know whether a situation is globally safe or not.

2.1.4 Conclusion of 2.1

Multi-body robot control laws in the literature are not safe: the avoidance method used do not take the robot capabilities into account. As a result, the system workspace is wasted and the behavior is dangerous. Intuitive expressions of constraints such as joint position limits and joint acceleration limits can become incompatible, which directly leads to a safety violation. As a result, before solving the control problem, it is mandatory to take the robot capabilities into account in the constraints expression and to study the potential incompatibilities between them to draw the boundaries of the space over which the control problem should be solved.

The assumptions of the study are the following:

1. The perception of the system is exact;

2. The models of the system and the environment are known exactly;

3. The system limits implemented in the controller are representative of the system real capabilities;

4. Once the control problem is expressed as *feasible* (all the constraints can be satisfied simultaneously), an algorithm is able to solve it without any constraint violation;

5. At a given time step, a desired control input (*e.g.* $\boldsymbol{u}(k) = \dot{\boldsymbol{q}}_{des}(k+1)$ for time step k) satisfying the constraints is perfectly carried out at the next time step ($\dot{\boldsymbol{q}}_{des}(k+1) = \dot{\boldsymbol{q}}(k+1)$).

The consequences of these assumptions are discussed for practical experiments in section 4.3.1.3.

2.2 Description for safety

This section is an interpretation based on the previous discussion: the definition of safety is introduced, based on an appropriate description of the system and its constraints.

2.2.1 E-state

First, it appears that the description of the behavior of a robotic system Σ and its constraints through its state $\boldsymbol{s}$ as defined in the State Representation formalism is insufficient. As a matter of fact, an extended state vector (e-state) is defined and denoted $\boldsymbol{\sigma}$; it gathers all the variables which allow to describe Σ and its constraints. The e-state is defined over continuous time ($t \in \mathbb{R}_+$) since it contains variables used to describe the physical system. For example, a n-DOF manipulator controlled at the velocity kinematic level and constrained by collisions avoidance and joint position, velocity and acceleration limits has the following e-state

$$\boldsymbol{\sigma} = \left[\boldsymbol{q}^T \ \dot{\boldsymbol{q}}^T \ \ddot{\boldsymbol{q}}^T \ \boldsymbol{d}^T\right]^T \tag{2.7}$$

where $\boldsymbol{d}^T$ is a vector of distances to obstacles[3]. In the same example, the state of Σ would be $\boldsymbol{s} = \boldsymbol{q}$. Conversely to $\boldsymbol{\sigma}$, the control vector $\boldsymbol{u}(k) = \dot{\boldsymbol{q}}(k+1)$ belongs to $\mathbb{R}^n$ and it is defined over the discrete time ($k \in \mathbb{N}$). The e-state space is denoted $\mathscr{S}$ and the control space ($\mathbb{R}^n$) is denoted $\mathscr{C}$.

2.2.2 E-state constraints

The notion of constraint usually refers to both a test on the system ("Is the joint boundary exceeded?"- denoted by *e-state constraint*) and a prerequisite to motion ("The control input sent to the actuator should not lead to exceed the joint boundary."- denoted by *control constraint*). The e-state constraints describe if Σ satisfies safety at the current time, *i.e.* when not considering any time horizon. They can be expressed through Boolean functions such as

$$\begin{aligned} &\mathscr{S} \rightarrow \mathscr{B} \\ f : \boldsymbol{\sigma} &\mapsto 1 \ \text{ if the constraint is satisfied} \\ &\quad\ 0 \ \text{ else,} \end{aligned} \tag{2.8}$$

where $\mathscr{S}$ is the e-state space and $\mathscr{B}$ the Boolean space.

[3]To avoid obstacles, various kind of distance computations can be retained: single or multiple distances for one robot segment or for one convex body (issued of a convex decomposition), use of activation areas, etc. In this example and in the rest of this work, this vector contains positive values and its size is dynamic.

As an example of e-state constraint, $f_{PM,3}$ (P for Position limit, M for Maximum) describes the superior position limit of the 3^{rd} joint

$$f_{PM,3} : \boldsymbol{\sigma}(t) \mapsto q_3(t) - q_{M,3} \leq 0 \tag{2.9}$$

where $q_3(t)$ is the joint position of joint 3 at time t and $q_{M,3}$ is the maximum joint boundary value.

p being in $\mathbb{N}$, any e-state satisfying the p e-state constraints imposed to Σ satisfies the property $\bigwedge_{i=1}^{p} (f_i(\boldsymbol{\sigma})) = 1$, where $\bigwedge$ is the logical conjunction operator (AND). This means that all e-state constraints are simultaneously true for the e-state $\boldsymbol{\sigma}$. In this case, $\boldsymbol{\sigma}$ is called an *instant-safe e-state*.

2.2.3 Subspaces of the e-state space and definition of safety

The e-state space $\mathscr{S}$ is composed of subspaces that can be identified. The subspace of $\mathscr{S}$ gathering all the instant-safe e-states is denoted $\mathscr{S}_A$. Conversely, the complementary subspace gathers the e-states violating an e-state constraint; it is denoted $\mathscr{S}_{VS}$ (VS for Violation e-State). As illustrated by Fig. 2.2, maintaining at each time step $\boldsymbol{\sigma}$ in $\mathscr{S}_A$ for the next time step is not sufficient to prevent an inevitable e-state constraint violation in the future. As a consequence, there is a subspace of $\mathscr{S}_A$ which should not be reachable by the system to guarantee its safety.

An e-state leading inevitably to an e-state constraint violation is called an Inevitable Violation e-State (IVS). It is an extension of the notion of Inevitable Collision State (ICS) defined by Fraichard *et al.* (2004) which denotes a state from which, whatever the sequence of control inputs sent, a collision finally occurs. Once an IVS is reached, the system can be considered as not safe anymore as an e-state constraint violation is going to happen. The space of IVS is a subspace of $\mathscr{S}_A$ denoted $\mathscr{S}_{IVS}$. The union of $\mathscr{S}_{IVS}$ and $\mathscr{S}_{VS}$ is denoted $\mathscr{S}_V$ and gathers all the e-states that should be avoided to ensure safety. The complement of $\mathscr{S}_V$ in $\mathscr{S}$ is denoted $\mathscr{S}_{\overline{V}}$. These subspaces are illustrated on Fig. 2.4. A definition of safety is then

Definition 2.2.3.1. *: Safety. The* ***safety*** *of a robotic system Σ is ensured at the control level if its e-state $\boldsymbol{\sigma}$ cannot reach $\mathscr{S}_V$.*

This definition enlightens the role of the constraints expression to limit the evolution of the system toward dangerous areas.

The last subspace to define in this section regards the space that is reachable by a system given its constraints expression. Given an initial e-state $\boldsymbol{\sigma}_0$, $\mathscr{R}(\boldsymbol{\sigma}_0)$ denotes the space of all the reachable e-states on an infinite time horizon through all the possible constraint compliant control (control law able to find a control vector compliant with the constraints whenever the control problem is feasible).

2.3 Methodology to study and ensure safety

This section exposes the methodology to ensure safety. The proposed methodology must be carried out offline, upstream from the robotic mission. The equivalent of e-state constraints should be formulated at the control level. Once their *validity* is ensured, they should either be proved *compatible*, or the permanent availability of an *alternative safe behavior* must be ensured on an infinite time horizon.

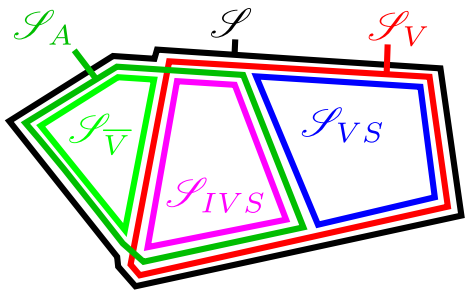

Figure 2.4: Partitioning of the e-state space. To be safe, a system should not be able to reach $\mathscr{S}_V$.

2.3.1 Step 1: Control constraints definition

The controller cannot act directly on the e-state $\boldsymbol{\sigma}$; it modifies it indirectly through the control vector $\boldsymbol{u}$. Inversely, at each time step, by imposing conditions on the e-state, each e-state constraint forbids an area of the control vector space $\mathscr{C}$. Hence, to each e-state constraint f is associated a control constraint F which can be defined as the function returning the space of *admissible* control vectors $\mathscr{C}_A(\boldsymbol{\sigma})$, *i.e.* the control vectors leading to an instant-safe e-state at the next time step. A control constraint can be expressed as a Boolean function returning whether a given control vector belongs to $\mathscr{C}_A(\boldsymbol{\sigma})$ or not

$$\begin{aligned} & (\mathscr{S},\mathscr{C}) \mapsto \mathscr{B} \\ F : (\boldsymbol{\sigma},\boldsymbol{u}) & \mapsto 1 \text{ if } \boldsymbol{u} \in \mathscr{C}_A(\boldsymbol{\sigma}) \\ & \quad\ 0 \text{ else.} \end{aligned} \tag{2.10}$$

The control input being discrete, control constraints are defined over discrete time ($k \in \mathbb{N}$). As an extension of the notation $\boldsymbol{\sigma}(t)$ ($t \in \mathbb{R}_+$), $\boldsymbol{\sigma}(k)$ ($k \in \mathbb{N}$) denotes the e-state at time step k.

In the example of the 3^{rd} joint superior position limit, if the control is done at the velocity kinematic level, a possible control constraint is

$$F_{PM,3} : \ (\boldsymbol{\sigma}(k), \dot{\boldsymbol{q}}(k+1)) \ \mapsto \dot{q}_3(k+1) - \frac{q_{M,3} - q_3(k)}{\delta t} \leq 0 \tag{2.11}$$

where δt is the time increment. It can be mentioned that from a practical point of view, the inequalities imposed on the system at each time step in the QP control law structure are an example of control constraints. At a given time step k, these terms are gathered in

$$J_C(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1) - \boldsymbol{b}(k) \leq 0. \tag{2.12}$$

2.3.2 Step 2: Validity

In order to ensure safety, the first stage is to check that control constraints are *valid*.

Definition 2.3.2.1. *: Validity. Let $\boldsymbol{\sigma}$ be an instant-safe e-state at time step k, a control constraint F is said valid if its satisfaction implies the satisfaction of its associated e-state constraint f at next time step $k+1$ and for all time between k and $k+1$.*

$$\boldsymbol{\sigma} \in \mathscr{S}_A, k \in \mathbb{N}, \forall t \in [k\delta t; (k+1)\delta t] :$$
$$F(\boldsymbol{\sigma}(k), \boldsymbol{u}(k)) = 1 \Rightarrow f(\boldsymbol{\sigma}(t)) = 1.$$

The validity of constraints is most of the time an assumption rather than a formally proven property. For example, constraints at various physical levels (position, velocity, acceleration, etc.) must be converted to the control physical level, which is often done thanks to first order approximations (finite differences). The control being in discrete time, the approximations induced by finite differences generate errors between the discrete ideal behavior and the real one. However, it is assumed that the sampling period is appropriately chosen to ensure that these errors remain acceptable with respect to the various usual sources of errors (model approximations, sensors precision, etc.). As a remark, it is always possible to find valid control constraints by reducing their space of admissible control vector $\mathscr{C}_A$.

2.3.3 Step 3: Compatibility

A second stage to ensure safety is to check that the set of control constraints is *compatible.*

Definition 2.3.3.1. *: Compatibility. Given an initial e-state $\boldsymbol{\sigma}_0$ in $\mathscr{S}_{\overline{V}}$, a set of p control constraints is compatible if for all e-state $\boldsymbol{\sigma}$ in $\mathscr{R}(\boldsymbol{\sigma}_0)$, there exists $\boldsymbol{u}$ in $\mathscr{C}$ such that $\bigwedge_{i=1}^{p} (F(\boldsymbol{\sigma}, \boldsymbol{u})) = 1$.*

The following proposition establishes that validity and compatibility ensure safety.

Proposition 2.3.1. *: Let $\boldsymbol{\sigma}$ in $\mathscr{S}_{\overline{V}}$ be the e-state of Σ, a robotic system constrained by p e-state constraints (p in $\mathbb{N}$). If the p control constraints of Σ are valid and compatible, then safety is ensured.*

Proof. : Let Σ be a robotic system in an initial (time step 0) e-state $\boldsymbol{\sigma}_0$ belonging to $\mathscr{S}_{\overline{V}}$. As the control constraints are compatible, there exists $\boldsymbol{u}$ in $\mathscr{C}$ such that $\bigwedge_{i=1}^{p} (F(\boldsymbol{\sigma}_0, \boldsymbol{u})) = 1$. Thus the control problem is feasible and as all the constraints are valid, the system e-state $\boldsymbol{\sigma}$ is instant-safe at time step 1 and for all time between time steps 0 and 1.

Similarly, as the control constraints are compatible, a system Σ having its e-state $\boldsymbol{\sigma}$ belonging to $\mathscr{S}_{\overline{V}}$ at time step n in $\mathbb{N}$ has a control vector $\boldsymbol{u}$ in $\mathscr{C}$ such that $\bigwedge_{i=1}^{p} (F(\boldsymbol{\sigma}, \boldsymbol{u})) = 1$. Thus the control problem is feasible and as all the constraints are valid, the system e-state $\boldsymbol{\sigma}$ is instant-safe at time step $n+1$ and for all time between time steps n and $n+1$.

By recursion, $\boldsymbol{\sigma}$ is maintained in $\mathscr{S}_A$ on an infinite time-horizon, which means that it is maintained in $\mathscr{S}_{\overline{V}}$; as a consequence, it cannot reach $\mathscr{S}_V$ and safety is ensured. □

2.3.4 Step 4: Design of Alternative Safe Behaviors

The study of compatibility between control constraints is complex: an exhaustive method would consist to, given an initial e-state $\boldsymbol{\sigma}_0$, evaluate all the control constraints for all the e-states $\boldsymbol{\sigma}$ reachable from $\boldsymbol{\sigma}_0$ to detect empty intersections between control admissibility spaces $\mathscr{C}_A(\boldsymbol{\sigma})$. Given the diversity of constraints, it seems vain to look for generic methods to detect incompatibilities and modify control constraints appropriately to eradicate

them. Moreover, sometimes incompatibilities cannot be resolved: when variables cannot be measured accurately, or when there is no model available, another method should be used to ensure safety.

A second way to guarantee safety is to ensure the permanent availability of a sequence of control solutions leading to instant-safe e-states on an infinite time horizon. At each time step, it is ensured that the controller will be able at next time step to switch to an infinite sequence of controls leading to exclusively instant-safe e-states. Similarly to the proof of proposition 2.3.1, $\boldsymbol{\sigma}$ is maintained in $\mathscr{S}_A$ on an infinite time-horizon, which means that it is maintained in $\mathscr{S}_{\overline{V}}$; as a consequence, it cannot reach $\mathscr{S}_V$ and safety is ensured. This control sequence is called an Alternative Safe Behavior (ASB - referred as evasive manoeuvres by Parthasarathi *et al.* (2007)). Dedicated ASBs are exposed in section 2.4.5 according to the specifications of the proposed case studies.

2.3.5 Summary and methodology

To describe the physical system Σ and its constraints in continuous time, the e-state $\boldsymbol{\sigma}$ is proposed, and the status of the system with respect to its constraints is given by the e-state constraints. Based on this description, a definition of safety at the control level is proposed: Σ is safe if its e-state $\boldsymbol{\sigma}$ is not able to reach the forbidden e-states $\mathscr{S}_V$. In order to prevent this, the control constraints F are defined, and the validity property keeps a link between control constraints in discrete time and e-state constraints in continuous time. To ensure safety, either the compatibility property must be proved for the set of control constraints, or the availability of an Alternative Safe Behavior must be ensured on an infinite time horizon.

The algorithm presented on Fig. 2.5 illustrates the following methodology:

1. Based on e-state constraints, formulate associated control constraints;
2. Prove validity;
3. Prove compatibility;
4. If step 3 is not possible, either go back to 1 and modify the control constraints expression or define a new control sequence among ASBs. These ASB will then be computed at each time step to ensure that the controller is able to switch at the following time step to an infinite sequence of controls leading to exclusively instant-safe e-states.

2.4 Safety preservation - case studies

As applications of the framework exposed above, three case studies are proposed in this section. The considered system is a n-DOFs manipulator controlled at the velocity kinematic level. The three case studies are associated to three constraints contexts exploited in the chapter 4:

- Joint position, velocity limits and static obstacles. This set of constraints is the context of an evolutionary design process which involves kinematic trajectory tracking simulations for evaluating manipulators morphologies (section 4.2);

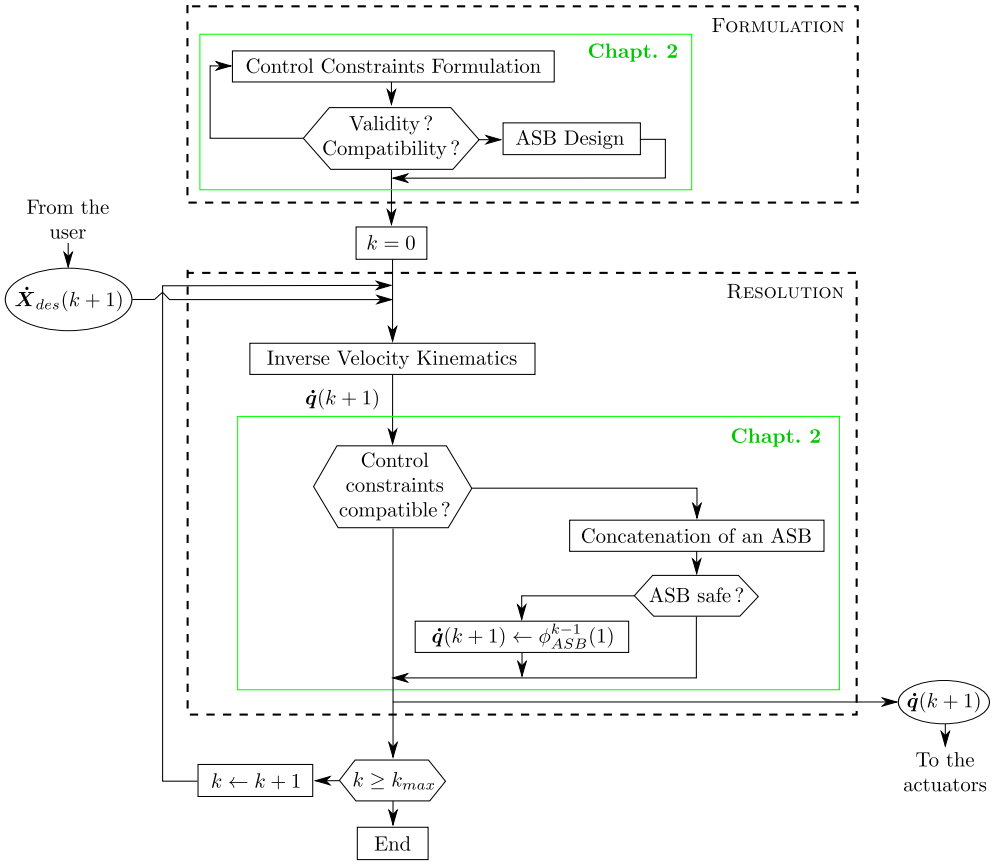

Figure 2.5: **Safe controller algorithm.** The part of the algorithm above the dashed line is offline and is concerned with the control problem formulation; the part of the algorithm under the dashed line is online and is concerned with the control problem resolution. From each identified e-state constraint (physical limit or induced by the mission), a control constraint is formulated offline. If the validity of the constraints cannot be proved, a new formulation of the control constraints must be expressed. It is always possible to find valid control constraints by reducing their space of admissible control vector $\mathscr{C}_A$. If the compatibility of the control constraints cannot be proved, a new formulation can be expressed and evaluated, or an ASB must be established. Once this is done, the reactive control loop is launched. At each time step, the controller is fed with operational inputs and solves the control problem thanks to any Constraints Compliant Control algorithm. In particular, it can include usual constraints avoidance techniques (*e.g.* the one of Maciejewski *et al* (1985)). If the compatibility of the control constraints defined offline could not be proved, an ASB sequence is concatenated to the desired joint motion: if the resulting behavior is not safe, then the first control input of the ASB is sent (represented by ϕ_{ASB}^{k-1}), which safety has been proved at a previous time step; else, the control solution is sent to the actuators.

- Joint position, velocity and acceleration limits. These constraints consider a real manipulator to be controlled in an empty environment; this case is simulated in section 4.1 and experimented in section 4.3;

- Joint position, velocity, acceleration limits and static obstacles. This set of con-

straints is used to control safely a real manipulator in a cluttered environment; this case is simulated in section 4.1 and experimented in section 4.3.

As a remark, moving obstacles and self-collisions (which can be considered as moving obstacles) are not considered in these case studies.

First, all the e-state constraints and preliminary expressions of all the control constraints are proposed. Then, the three case studies are exposed.

2.4.1 E-state constraints expression

The e-state constraints expressions specify if an e-state $\boldsymbol{\sigma}$ of the system Σ is instant-safe at a given time. The following equations are expressed for a given time t in $\mathbb{R}_+$.

- **Joint position limit**

$$f_{PM} : \boldsymbol{\sigma} \mapsto \boldsymbol{q}(t) \leq \boldsymbol{q_M} \tag{2.13}$$
$$f_{Pm} : \boldsymbol{\sigma} \mapsto \boldsymbol{q_m} \leq \boldsymbol{q}(t) \tag{2.14}$$

  where $\boldsymbol{q_m}$ and $\boldsymbol{q_M}$ are respectively the minimum and the maximum joint position limits.

- **Joint velocity limit**

$$f_{VM} : \boldsymbol{\sigma} \mapsto \dot{\boldsymbol{q}}(t) \leq \dot{\boldsymbol{q}}_{\boldsymbol{M}} \tag{2.15}$$
$$f_{Vm} : \boldsymbol{\sigma} \mapsto \dot{\boldsymbol{q}}_{\boldsymbol{m}} \leq \dot{\boldsymbol{q}}(t) \tag{2.16}$$

  where $\dot{\boldsymbol{q}}_{\boldsymbol{m}}$ and $\dot{\boldsymbol{q}}_{\boldsymbol{M}}$ are respectively the minimum and the maximum joint velocity limits. All elements of $\dot{\boldsymbol{q}}_{\boldsymbol{m}}$ are negative; all elements of $\dot{\boldsymbol{q}}_{\boldsymbol{M}}$ are positive.

- **Joint acceleration limit**

$$f_{AM} : \boldsymbol{\sigma}(t) \mapsto \ddot{\boldsymbol{q}}(t) \leq \ddot{\boldsymbol{q}}_{\boldsymbol{M}} \tag{2.17}$$
$$f_{Am} : \boldsymbol{\sigma}(t) \mapsto \ddot{\boldsymbol{q}}_{\boldsymbol{m}} \leq \ddot{\boldsymbol{q}}(t) \tag{2.18}$$

  where $\ddot{\boldsymbol{q}}_{\boldsymbol{m}}$ and $\ddot{\boldsymbol{q}}_{\boldsymbol{M}}$ are respectively the minimum and the maximum joint acceleration limits. All elements of $\ddot{\boldsymbol{q}}_{\boldsymbol{m}}$ are negative; all elements of $\ddot{\boldsymbol{q}}_{\boldsymbol{M}}$ are positive.

- **Collisions avoidance** A collision is characterized by

$$\Sigma \bigcap \Omega \neq \emptyset \tag{2.19}$$

  where Σ is the system (meant here as the set of all the system points) and Ω is the set of all the obstacles points. The e-state constraint expression is then

$$f_O : \ \boldsymbol{\sigma} \ \mapsto \forall A \in \Sigma, G_A(\boldsymbol{q}) \notin \Omega \tag{2.20}$$

  where $G_A(\boldsymbol{q})$ is the geometric model of point A belonging to the robot.

These expressions are voluntarily simple and may not be sufficient to describe precisely real situations. For example, a joint velocity limit or a joint acceleration limit depends on the configuration and on the velocities of all the system joints for most systems. In the particular case of virtual manikins, even the joint position limits depend on the configuration. However, the proposed e-state expressions are considered sufficient in approximate cases (using an under-estimation of the real acceleration capabilities for example) and their simplicity is relevant to illustrate the methodology.

2.4.2 Case study 1: Joint position and velocity limits, Collisions avoidance

This case study involves three constraints: joint position limits, joint velocity limits and collisions avoidance. The following control constraints (assumed to be valid, see section 2.3.2) are derived from Eqs. (2.13) - (2.16) and (2.20) thanks to finite differences.

Joint position limit (ith joint) F_P

$$F_{PM,i} :(\boldsymbol{\sigma}, \boldsymbol{u}(k))\mapsto J_{c_i^+}\dot{\boldsymbol{q}}(k+1) \leq \frac{q_{M,i} - q_i(k)}{\delta t} \tag{2.21}$$

$$F_{Pm,i} :(\boldsymbol{\sigma}, \boldsymbol{u}(k))\mapsto J_{c_i^-}\dot{\boldsymbol{q}}(k+1) \leq \frac{q_i(k) - q_{m,i}}{\delta t} \tag{2.22}$$

where $J_{c_i^+} = [0, \ldots, 0, 1, 0, \ldots, 0]$ (the i^{th} term being 1) and $J_{c_i^-} = [0, \ldots, 0, -1, 0, \ldots, 0]$.

Joint velocity limit (ith joint) F_V

$$F_{VM,i} :(\boldsymbol{\sigma}(k), \boldsymbol{u}(k))\mapsto J_{c_i^+}\dot{\boldsymbol{q}}(k+1) \leq \dot{q}_{M,i} \tag{2.23}$$

$$F_{Vm,i} :(\boldsymbol{\sigma}(k), \boldsymbol{u}(k))\mapsto J_{c_i^-}\dot{\boldsymbol{q}}(k+1) \leq -\dot{q}_{m,i}. \tag{2.24}$$

Collisions avoidance F_O

$$F_O :(\boldsymbol{\sigma}, \boldsymbol{u}(k))\mapsto J_{A,B}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1) \leq \frac{d_{A,B}(k)}{\delta t} \tag{2.25}$$

for all pairs of point (A, B), where A belongs to the robot and B to the obstacles; $d_{A,B}$ is the distance between A and B; $J_{A,B}(\boldsymbol{q}(k))$ is the (line) Jacobian of point A along the direction $A \rightarrow B$. For practical reasons, this infinite set of constraints is reduced to one constraint per segment of the robot (shortest distance to the environment). This technique is frequently used despite its limits in some cases, as shown by Kanehiro *et al.* (2008). Another method proposed by Peinado *et al.* (2009) consists in assessing analytically the system configuration in its environment to estimate the areas subject to a potential collision and act adequately through an iterative process. This approach builds incrementally collision-free solutions and avoids the approximations induced by the linearization of collisions constraints (as when using of $J_{A,B}$). In the present study, the proposed expression is considered sufficient. As mentionned at the beginning of section 2.4, this expression is limited to the description of static obstacles (no moving obstacles of self collisions).

The space of admissible control vectors for control constraints F_P, F_V and F_O are respectively denoted $\mathscr{C}_A^P$, $\mathscr{C}_A^V$ and $\mathscr{C}_A^O$.

Validity being assumed (cf. section 2.3.2), the compatibility is checked.

Proposition 2.4.1. *: The set $\{F_P, F_V, F_O\}$ is compatible*

Proof. : Let $\dot{\boldsymbol{q}}_0$ be the null control vector ($\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$) and let $\boldsymbol{\sigma}_0$ be in $\mathscr{S}_{\overline{V}}$. For any $\boldsymbol{\sigma} \in \mathscr{R}(\boldsymbol{\sigma}_0)$, $\dot{\boldsymbol{q}}(k+1) = \dot{\boldsymbol{q}}_0$ belongs to $\mathscr{C}_A^P$, $\mathscr{C}_A^V$ and $\mathscr{C}_A^O$. As a result, for all $\boldsymbol{\sigma}(k)$ in $\mathscr{R}(\boldsymbol{\sigma}_0)$, $\dot{\boldsymbol{q}}_0$ is solution of the control problem and thus $F_P(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) \bigwedge F_V(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) \bigwedge F_O(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) = 1$. □

The compatibility of this set is illustrated on Fig. 2.6. This figure describes the safety of a system submitted to various constraints. For example, the cell "12" stands for a system submitted to F_P (denoted by "1") and F_V (denoted by "2"); as it is written in black, the system is safe. As all the control constraints are proved to be always compatible, safety is ensured without modification or ASB required.

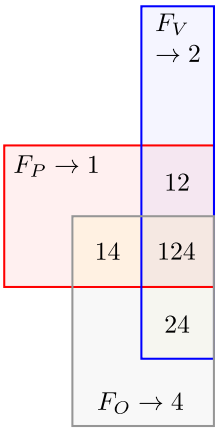

Figure 2.6: Subsets generated by the set of constraints $\{F_P, F_V, F_O\}$. F_P, F_V and F_O are respectively represented by numbers 1, 2 and 4. Regarding the possible combination of F_P, F_V and F_O, as these control constraints are compatible, safety is ensured.

2.4.3 Case study 2: Joint position, velocity and acceleration limits

As in the previous case, this case study involves three constraints but collisions avoidance is replaced by joint acceleration limits. The control constraints for joint position and velocity limits are taken from case study 1 (Eqs. (2.21) - (2.24)); the control constraint of joint acceleration limit is derived from Eqs. (2.17) and (2.18) thanks to finite differences (i^{th} joint)

$$F_{AM,i} : (\boldsymbol{\sigma}(k), \boldsymbol{u}(k)) \mapsto \quad (2.26)$$
$$J_{c_i^+} \dot{\boldsymbol{q}}(k+1) \leq \ddot{q}_{M,i}\delta t + \dot{q}_i(k)$$
$$F_{Am,i} : (\boldsymbol{\sigma}(k), \boldsymbol{u}(k)) \mapsto \quad (2.27)$$
$$J_{c_i^-} \dot{\boldsymbol{q}}(k+1) \leq -\ddot{q}_{m,i}\delta t - \dot{q}_i(k).$$

The space of admissible control vectors for control constraint F_A is denoted $\mathscr{C}_A^A$.

As illustrated by Fig. 2.7, the preliminary constraints expressions of joint position limits and acceleration limits are incompatible.

Proposition 2.4.2. *The sets of control constraints generated by $\{F_P, F_A\}$ are incompatible.*

Proof. : let $\boldsymbol{\sigma}_0$ be in $\mathscr{S}_{\overline{V}}$. From $\boldsymbol{\sigma}_0$, any $\boldsymbol{\sigma}_V \in \mathscr{R}(\boldsymbol{\sigma}_0)$ for which a given joint satisfies

$$\dot{q}(k) > \frac{q_M - q(k)}{\delta t} - \ddot{q}_m \delta t \quad (2.28)$$

is such that $F_{PM,i}(\boldsymbol{\sigma}_V)$ and $F_{Am,i}(\boldsymbol{\sigma}_V)$ are not compatible, which traduces that $f_{PM,i}$ and $f_{Am,i}$ cannot be satisfied simultaneously. As there is no assumption or constraint preventing from reaching $\boldsymbol{\sigma}_V$, then F_P and F_A are incompatible. □

This incompatibility is illustrated on Fig. 2.2. It has been locally treated by Decré *et al.* (2009), but as shown in appendix A, the proposed method is tight and can be smoothened by imposing that the joint distance to the joint position limit at next time

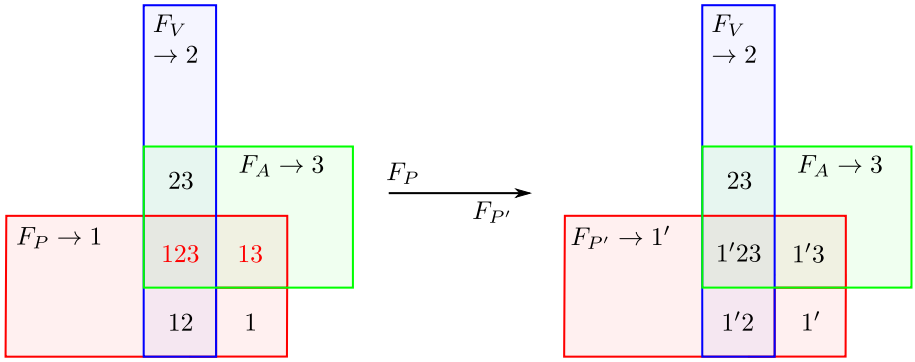

Figure 2.7: Illustration of the compatibility of all the constraints subsets of case study 2. F_P, $F_{P'}$, F_V, F_A and F_O are respectively represented by numbers 1, 1', 2, 3 and 4. The initial control constraints of joint position and acceleration limits are incompatible, which causes incompatibilities in all the sets containing both (in red). The modification of the joint position limit control constraint ensures the control constraints compatibility.

step should remain superior to the current joint distance needed to decelerate. As a result, a modified expression of F_P is proposed (i^{th} joint)

$$F_{PM',i} : (\boldsymbol{\sigma}, \boldsymbol{u}(k)) \mapsto \tag{2.29}$$
$$J_{c_i^+}\dot{q}(k+1) \leq \frac{(q_M - q(k)) - \frac{1}{2}(s_1^2 - s_1)\ddot{q}_m\delta t^2}{(s_1+1)\delta t}$$
$$F_{Pm',i} : (\boldsymbol{\sigma}, \boldsymbol{u}(k)) \mapsto \tag{2.30}$$
$$J_{c_i^-}\dot{q}(k+1) \leq \frac{(q_m - q(k)) - \frac{1}{2}(s_2^2 - s_2)\ddot{q}_M\delta t^2}{(s_2+1)\delta t}$$

with

$$s_1 = -\frac{\sqrt{-2\ddot{q}_m(q_M - q(k))}}{\ddot{q}_m\delta t}, \tag{2.31}$$
$$s_2 = \frac{\sqrt{-2\ddot{q}_M(q_m - q(k))}}{\ddot{q}_M\delta t}. \tag{2.32}$$

Proposition 2.4.3. *The set* $\{F_{P'}, F_V, F_A\}$ *is compatible.*

Proof. Let $\boldsymbol{\sigma}_0$ be in $\mathscr{S}_{\overline{V}}$, the current time step k be in $\mathbb{N}$ and the current e-state $\boldsymbol{\sigma}(k)$ be in $\mathscr{R}(\boldsymbol{\sigma}_0)$. The design of Eqs. (2.29) and (2.30) is based on the condition

$$\boldsymbol{\Delta q}(k+1) > \boldsymbol{d}_{\mathbb{R},dec}(k) \tag{2.33}$$

where $\boldsymbol{\Delta q}(k+1)$ is the joint distance to the position limit at the next time step and $\boldsymbol{d}_{\mathbb{R},dec}(k)$ is a vector of upper bounds of the joint distances needed to stop at current time step (cf. appendix A). This condition implies that the vector of maximum deceleration velocity $\dot{\boldsymbol{q}}_{dec}(k+1)$ which general term is

$$\dot{q}_{dec}(k+1) = \begin{cases} \dot{q}(k) + \ddot{q}_m\delta t & \text{if } \dot{q}(k) \geq -\ddot{q}_m\delta t \\ \dot{q}(k) + \ddot{q}_M\delta t & \text{if } \dot{q}(k) \leq -\ddot{q}_M\delta t \\ 0 & \text{else} \end{cases} \tag{2.34}$$

belongs to $\mathscr{C}_A^{P'}(\boldsymbol{\sigma})$. Then, by definition, it belongs to $\mathscr{C}_A^{A}(\boldsymbol{\sigma})$. Finally, as it reduces the velocity magnitude, it belongs to $\mathscr{C}_A^{V}(\boldsymbol{\sigma})$. As a result, for all $\boldsymbol{\sigma}(k)$ in $\mathscr{R}(\boldsymbol{\sigma}_0)$, $\dot{\boldsymbol{q}}_{dec}$ is solution of the control problem and thus $F_{P'}(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) \bigwedge F_V(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) \bigwedge F_A(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_0) = 1$. □

This incompatibility between control constraints being resolved, the control constraints are ensured to be always compatible, which ensures safety.

2.4.4 Case study 3: Joint position, velocity and acceleration limits, Collisions avoidance

This case study involves four constraints, gathering the two previous case studies: joint position limits, joint velocity limits, joint acceleration limits and collisions avoidance. The considered control constraints are Eqs. (2.23) - (2.27), (2.29) and (2.30).

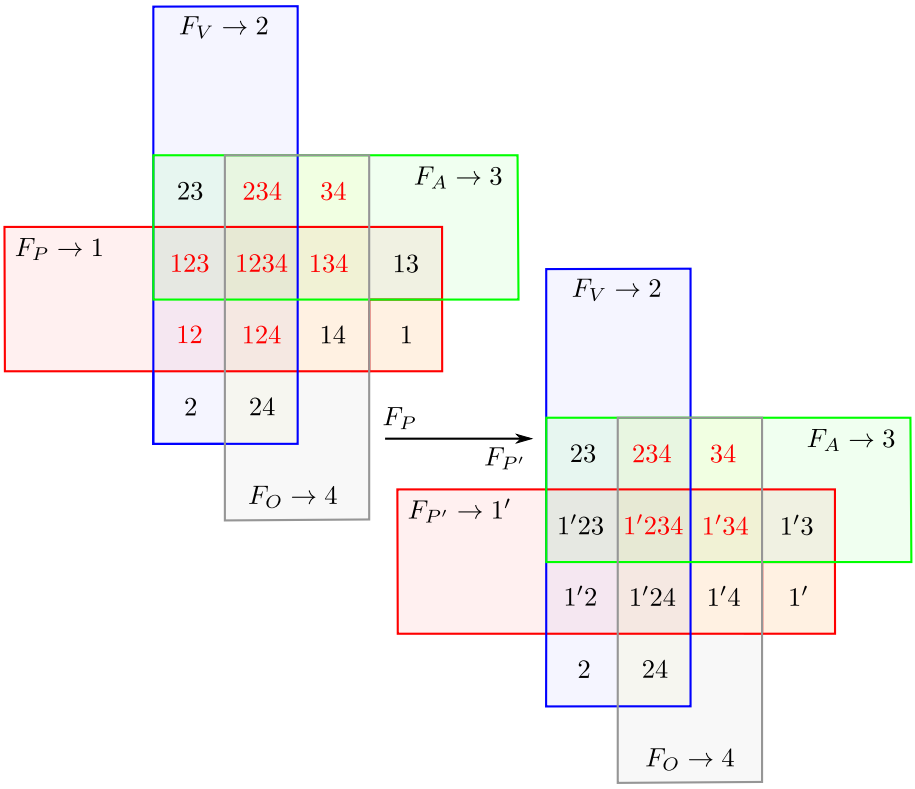

Figure 2.8: Illustration of the compatibility of all the constraints subsets of case study 3. F_P, $F_{P'}$, F_V, F_A and F_O are respectively represented by numbers 1, 1', 2, 3 and 4. The modifications brought to joint position limits control constraint avoid incompatibilities between joint position and joint acceleration limits. However, incompatibilities between joint acceleration limits and obstacles remain.

Proposition 2.4.4. *: The sets generated by $\{F_O, F_A\}$ are incompatible.*

Proof. : Let $\boldsymbol{\sigma}_0$ be in $\mathscr{S}_V$. From $\boldsymbol{\sigma}_0$, any $\boldsymbol{\sigma}_V \in \mathscr{R}(\boldsymbol{\sigma}_0)$ for which

$$J_{A,B}(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k) > \frac{d_{A,B}(k)}{\delta t} - J_{A,B}(\boldsymbol{q}(k))\ddot{\boldsymbol{q}}_m \delta t \tag{2.35}$$

shows that $F_O(\boldsymbol{\sigma})$ and $F_{Am}(\boldsymbol{\sigma})$ are not compatible, which shows that $f_O(\boldsymbol{\sigma})$ and $f_{Am}(\boldsymbol{\sigma})$ cannot be satisfied simultaneously. As there is no assumption or constraint preventing from reaching $\boldsymbol{\sigma}_V$, then F_O and F_A are incompatible. □

The incompatibility induced by the simultaneous presence of F_O and F_A is complex. Actually, the operational acceleration depends on the robot configuration (derived from Eq. (2.25)).

$$\ddot{X}_{A,B} = J_A(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.36}$$

which does not enable to rely on any value for the operational acceleration capabilities along a trajectory. In the worst cases, these capabilities may fall down to zero, which prevents to take a lower bound on which to rely for the deceleration capabilities estimation. As a result, ensuring compatibility between joint acceleration limits and collisions avoidance seems impossible without an exploration in the neighborhood of the current system e-state, which may turn time-consuming and thus not acceptable in real-time reactive control. In this case, the permanent availability of an Alternative Safe Behavior is required.

2.4.5 Alternative Safe Behaviors

When the control compatibility cannot be proved, the permanent availability of an Alternative Safe Behavior is required, to be triggered in case of critical situation. As mentioned in section 2.3.4, an ASB is a sequence of control solutions $\boldsymbol{u}$ leading to instant-safe e-states on an infinite time horizon. It must be computed at each time step and the safety of the resulting e-states must be checked, thus it should be fast to compute. An ASB is a restriction of the control problem to a unique possible solution; this can be seen as an Emergency Stop at the control level. It ensures the constraints satisfaction but neglects the operational objectives. This may be detrimental to the tasks under progress (orientations or positions of tools may be modified for example) but these potential drawbacks are considered as secondary with respect to a constraint violation.

To clarify the following descriptions, let ϕ denotes an infinite constraint compliant control input, *i.e.* an infinite sequence of controls $\boldsymbol{u}$ satisfying the control constraints at each time step.

2.4.5.1 Algorithm based on maximum joint deceleration ASB

As a preliminary observation, as the environment is assumed to be static, once an instant-safe e-state is static (no variation with respect to time), it remains safe until the end of time. Consequently, the first ASB proposed ϕ_{ASB1} is a full deceleration at the joint level. This deceleration is the most efficient way to stop the robot: it is fast as no Jacobian has to be recomputed at each time step, and the number of time steps necessary to obtain a static robot is minimized. As the control constraints of joint position, velocity and acceleration limits are compatible, the only remaining constraint to check on all e-states resulting from ϕ_{ASB1} is F_O, that is an intersection between the robot bodies and the environment. The method is detailed on algorithm 1. It is assumed that for $k = 0$, the initial e-state $\boldsymbol{\sigma}_0$ belongs to $\mathscr{S}_{\overline{V}}$. The algorithm is illustrated on Fig. 2.9.

Algorithm 1 : Maximum joint deceleration ASB

for all $k \in \mathbb{N}$ **do**
  (1) Compute an admissible solution $\dot{\boldsymbol{q}}(k+1)$
  (2) Compute deceleration trajectory ϕ^k_{ASB1}
  if (3) for all $\boldsymbol{\sigma}$ resulting from ϕ^k_{ASB1}, $\boldsymbol{\sigma}$ is an instant-safe e-state **then**
    (4.1) Send $\dot{\boldsymbol{q}}(k+1)$
  else
    (4.2) Send the first element of ϕ^{k-1}_{ASB1}
  end if
end for

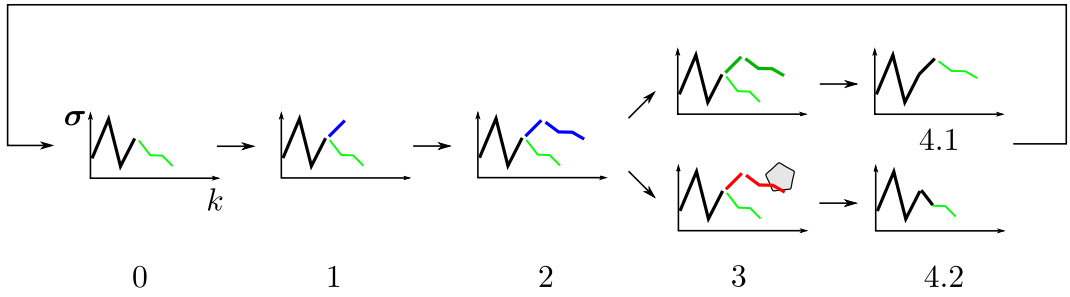

Figure 2.9: Algorithm of maximum deceleration based ASB. *Blue* is for non validated motion, *red* is for non admissible motion and *green* for safe motion. In *thin green*, the ASB computed at the previous time step. 0/ Situation at the beginning of the time step; 1/ Control solution computation $\dot{\boldsymbol{q}}(k+1)$; 2/ ASB1 profile computation ϕ^k_{ASB1}; 3/ Admissibility check; 4.1 (up) and 4.2 (down)/ Send appropriate output.

2.4.5.2 Algorithm based on mixable joint deceleration ASB

The algorithm of maximum joint deceleration ASB triggers a full joint deceleration at the very last moment, when the next computed motion (concatenated with a prediction of full joint deceleration) would lead to a collision. As a consequence, this algorithm may show low performances when the robotic system works near obstacles. The problem lies in the *maximal* deceleration toward the stop e-state; when ϕ_{ASB1} is chosen at one time step, it is likely to be applied until the robot is completely stopped very close to the environment. Then, the robot may oscillate between two behaviors when moving along the environment: 1/ computation of a "nominal" $\dot{\boldsymbol{q}}(k+1)$ issued from the operational input treated by the control law, and 2/ ϕ^{k-1}_{ASB1} issued from the ASB, as the proximity to obstacles is likely to require the ASB.

As shown on a simple example in Fig. 2.10, in most cases when a maximum deceleration ASB ϕ_{ASB1} has begun, there is no other possibility than full deceleration until the complete robot stop. To get a small margin in the control admissibility space during the deceleration, it is then proposed to impose the resort to an ASB earlier, so that the braking motion is not critical and the controller can allow other behaviors than full deceleration. To that end, the use of a prediction ϕ_{ASB2} with reduced acceleration capabilities is proposed.

At each time step, once the control solution $\dot{\boldsymbol{q}}(k+1)$ has been computed by the controller through the "nominal" control law, each prediction (ϕ_{ASB1} and ϕ_{ASB2}) is concatenated separately to the current control solution. Then, 4 possibilities may occur,

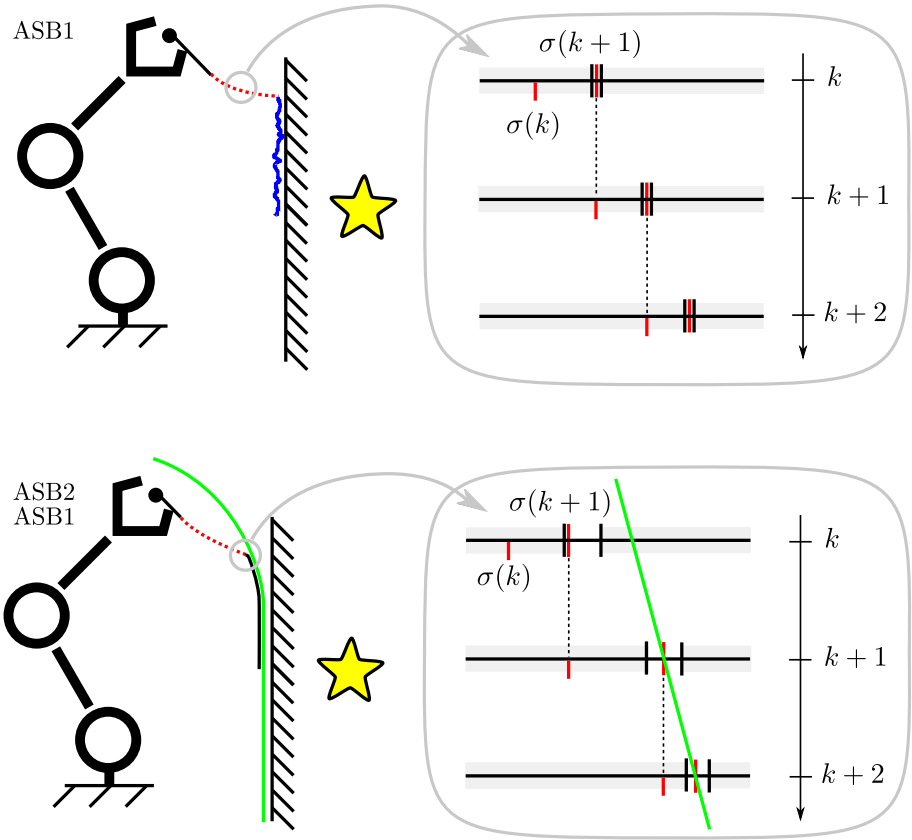

Figure 2.10: Comparative behaviors of robots trying to reach a keypoint (star) behind a wall. On the right, the schemes are representations of the e-states projected on the joint space of the 2^{nd} DOF of the system during 3 time steps. **Top**: maximum joint deceleration (ASB1). The motion of the robot is decomposed in three parts. Black path: the motion is computed through the control law, and at each time step the controller concatenates the control vector to be sent with a full deceleration, to check if a collision occurs and decide if the control vector should be sent or not. Red dashed path: a collision with the predicted full deceleration being detected, it is applied before sending the control law computed input; during the ASB deceleration, only the full deceleration control solution is admissible (top right). However, when the robot stops, it is close to the obstacle. Blue path: once near the obstacles, the controller oscillates between the control law solutions and ASB. **Bottom**: mixable joint deceleration (ASB1 & ASB2). As in the scheme at the top, the motion of the robot is decomposed in three parts. Black path: control law based motion; it is shorter than the upper one, because deceleration predictions are based on under-estimated capabilities. Red dashed path: the ASB is done with maximal deceleration capabilities, but as it has been triggered before, the control law solutions can be chosen in a small (but not reduced to a point) interval (bottom right). Black path on the green curve: as a result, the robot progression toward the wall can be damped by a smooth path constraint. Green curve: representation of the smooth path trajectory.

3 of them leading to the same action:

- no collision is encountered, neither for ϕ_{ASB1}, nor for ϕ_{ASB2}. The e-state $\sigma_{\dot{\boldsymbol{q}}(k+1)}$

resulting from the control solution $\dot{\boldsymbol{q}}(k+1)$ belongs to $\mathscr{S}_{\overline{V}}$, and it can be sent safely to the actuators;

- ϕ_{ASB1} encounters a collision and ϕ_{ASB2} not. It is the case met in the maximum joint deceleration ASB, $\sigma_{\dot{\boldsymbol{q}}(k+1)}$ may belong to $\mathscr{S}_V$ and thus ϕ^{k-1}_{ASB1} should be sent to the actuators for safety reasons.

- ϕ_{ASB1} and ϕ_{ASB2} encounter a collision. Similarly to the previous point, as ϕ_{ASB1} encounters a collision, it is the case met in the maximum joint deceleration ASB and ϕ^{k-1}_{ASB1} should be sent to the actuators for safety reasons.

- ϕ_{ASB2} encounters a collision and ϕ_{ASB1} not. This case is more likely to happen than the 2 previous ones as ϕ_{ASB2} is generated with reduced deceleration capabilities. This case traduces that the robot is close to an unavoidable collision but some margin (in the control admissibility space) remains. This free space can be exploited to insert during the incoming time steps an appropriate behavior to the robot, i.e. a behavior that would smoothens its reactions close to the environment. This exploitation is exposed hereafter; to maintain the margin, a full deceleration ϕ^{k-1}_{ASB1} is sent to the actuators.

The behavior to be inserted is actually induced by a dedicated control constraint F_C which is valid but not *a priori* compatible with the other constraints (else, if such a control constraint would be known, there would not be necessary to use an ASB). It is designed so that its satisfaction induces good performances of the system (in terms of smoothness for example), but its violation must be considered as acceptable, as its addition in the set of constraints may not produce a feasible control problem; in that case, F_C not considered in the control problem.

As an example for F_C, the method proposed by Faverjon and Tournassoud (1987) (referred later as Smooth Avoidance Technique (SAT)) can be used profitably. Briefly, this method constrains the operational velocities of each point of the robot bodies that gets close to an obstacle; each velocity limitation is done through an inequality constraint and the whole control problem is solved by a QP algorithm (Eq. (2.6)). The general velocity limit expression is

$$\dot{d} = -a\frac{d - d_s}{d_i - d_s} \qquad \text{for } d \leq d_i, \tag{2.37}$$

where $\dot{d}$ is the temporal derivative of the distance d between the robot point and the obstacle. This technique involves 3 parameters: a is a positive coefficient for adjusting convergence speed, d_s is the security distance (envelope) and d_i is the distance of influence, *i.e.* the distance under which the constraint is activated. A control constraint can be derived from Eq. (2.37)

$$F_C : \ (\boldsymbol{\sigma}(k), \boldsymbol{u}(k)) \ \mapsto J_A\dot{\boldsymbol{q}}(k) \leq -a\frac{d_{A,B}(k) - d_s}{d_i - d_s} \ \text{ for } d_{A,B}(k) \leq d_i. \tag{2.38}$$

As a remark, checking at each time step the feasibility of a control problem may not be trivial: knowing if a set of linear constraint is compatible may require the resolution of the associated linear system. When the set of considered constraint is $\{F_P, F_V, F_A, F_O, F_C\}$, an approximate answer can be given by checking whether the

configuration of maximum deceleration is admissible. It is not a requirement for compatibility (there may be cases for which this configuration is not admissible whereas the constraints are compatible) but it is a sufficient condition. As a result, at each time step the compatibility between the SAT control constraint and the other constraints is checked: if the SAT is not compatible, it is not considered.

The final method is exposed on algorithm 2 and detailed on Fig. 2.11 (step by step illustration). As for algorithm 1, it is assumed that for $k = 0$, the initial e-state $\boldsymbol{\sigma}_0$ belongs to $\mathscr{S}_{\overline{V}}$.

Algorithm 2 : Mixable joint deceleration ASB

for all $k \in \mathbb{N}$ **do**
  Control constraints: $\{F_P, F_V, F_A, F_O, F_C\}$
  if $\bigwedge_{i=1}^{n_{cc}} (F(\boldsymbol{\sigma}, \dot{\boldsymbol{q}}_{\boldsymbol{dec}}(k+1))) \neq 1$ **then**
    Control constraints: $\{F_P, F_V, F_A, F_O\}$
  end if
  (1) Compute an admissible solution $\dot{\boldsymbol{q}}(k+1)$
  (2) Compute deceleration trajectories ϕ^k_{ASB1}, ϕ^k_{ASB2}
  if (3) for all $\boldsymbol{\sigma}_1$ resulting from ϕ^k_{ASB1}, $\boldsymbol{\sigma}_1$ is a safe e-state AND for all $\boldsymbol{\sigma}_2$ resulting from ϕ^k_{ASB2}, $\boldsymbol{\sigma}_2$ is a safe e-state **then**
    (4.1) Send $\dot{\boldsymbol{q}}(k+1)$
  else
    (4.2) Send the first element of ϕ^{k-1}_{ASB1}
  end if
end for

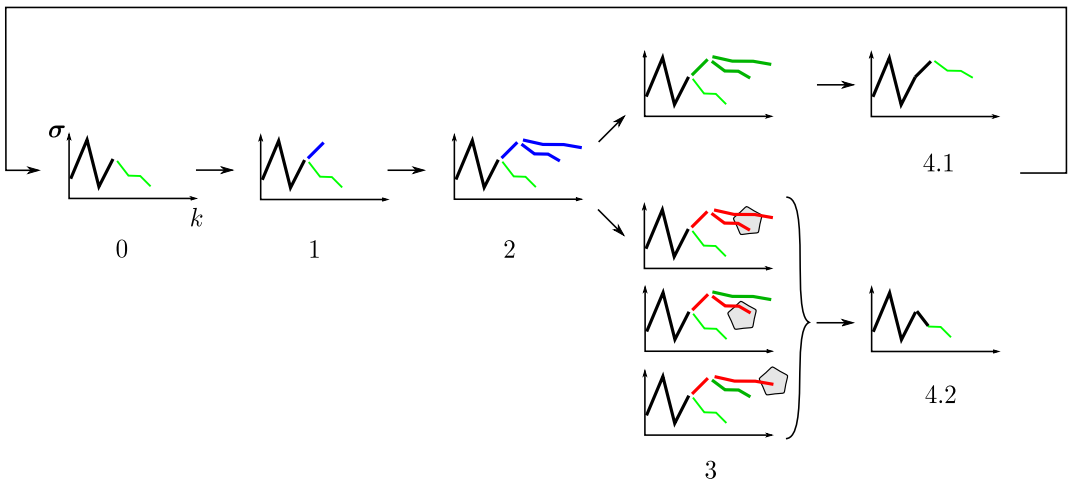

Figure 2.11: Algorithm of mixable deceleration based ASB. *Blue* is for non validated motion, *red* is for non admissible motion and *green* for safe motion. In *thin green*, the ASB computed at the previous time step. 0/ Situation at the beginning of the time step; 1/ Control solution computation $\dot{\boldsymbol{q}}(k+1)$; 2/ ASB1 ($\phi^k_{ASB1}$) and ASB2 ($\phi^k_{ASB2}$) profile computation; 3/ Admissibility check; 4.1 (up) and 4.2 (down)/ Send appropriate output.

There is no theoretical proof that the SAT always smoothens the robot behavior and prevents oscillations. However, as an informal clue, it can be mentioned that a robot

moving slowly along an obstacle has a small velocity projection along the axis toward the obstacle. If the problem is considered decoupled as a rough approximation (motion along the wall are considered independent to motion toward the wall), the SAT has only to deal with the part of the motions toward obstacles. The technique of Faverjon *et al.* (1987) is designed to manage such cases; the additional work on compatibility enables to reduce significantly the envelope d_s while ensuring safety. Finally, it is important to recall that the main goal of an ASB is to ensure safety at all costs; as a drawback, an ASB decreases the quality of solutions in terms of objectives tracking. The ASB should be used as rarely as possible; it should remain a kind of emergency solution in control.

2.5 Partial conclusion and perspectives

This chapter is dedicated to the reactive control problem formulation to obtain a safe problem behavior. It is based on the safety criteria enounced by Fraichard (2007).

A framework is exposed in which safety is expressed as the conservation of the system e-state in an admissibility space. This conservation is obtained through the compatibility of the control constraints (expressions to be satisfied at each time step by the control law). This compatibility can be ensured either by making the control constraints compatible, or by ensuring that an Alternative Safe Behavior is available at each time step.

The use of this framework is illustrated by case studies involving a n-DOF manipulator controlled at the velocity kinematic level; the considered constraints are joint position, velocity and acceleration limits and static obstacles. Examples of Alternative Safe Behavior are proposed to obtain a safe but also acceptable robot behavior.

Many perspectives are foreseen:

- Application to other constraints: joint torque or power limits, moving obstacles, etc. The exploitation of recent work based on Inevitable Collision States can be of much interest (Fraichard *et al.* 2004, Parthasarathi *et al.* 2007, Martinez-Gomez *et al.* 2008, 2009, Bautin *et al.* 2010). As a remark, dealing with moving obstacles is particularly complex in our case: as the constraints compatibility depends on the prediction of the system motion capabilities with respect to the environment, it strongly depends on the assumptions over the environment dynamics. A first perspective in that field is the management of self-collisions, which can be considered as a particular case of mobile obstacles with known dynamics;

- Application to controllers at other physical levels, such as position or torque control;

- Exploitation of the Partial Motion Planning studies (Petti *et al.* 2005, Van Den Berg *et al.* 2006, Bertolazzi *et al.* 2007, Hauser 2011). Most studies are limited to single body mobile robots, but even if the planning for multi-body robots is time consuming, real-time planning is likely to be more and more used in robotics. Research for intelligent ASBs is closely linked to this domain.

Chapter 3

Control problem resolution

The *control problem* usually covers various domains: planning, artificial intelligence, actuators low level loop... The scope of this chapter is the mid-level loop resolution: it is assumed that 1/ an operational input is given at each time step, coming from a high level loop, 2/ a joint output should be provided at each time step, going to the actuators low level loop. The proposed resolution method is based on the problem formulation elaborated in chapter 2: it is assumed that the problem is solvable. This is not generally the case in the literature, where the constraints are usually taken with sufficient margins to compensate for the possible incompatibilities.

According to the need of the user, the tasks specified to the robotic system are expressed in different ways in the literature: with the same or various (potentially strict) priorities, by equalities (desired value) or inequalities (desired areas). It appears that there are two main resolution families generally used to solve the control problem: 1/ Analytic Inversion Methods (AIM) which focus on the model inversion to obtain a solution in a single resolution iteration; 2/ Convex Optimization Methods (COM) which resort to all the available tools of this class of the mathematical optimizations field. There is much interest to combine the comprehension of the robotics issues through the AIM methods (widely used in robotics since the beginning of the exploitation of redundancy by Liégeois (1977)) and the efficiency of the COM tools (used less frequently but with much benefit, *e.g.* Faverjon *et al.* (1987), Escande *et al.* (2010)). A special attention is paid in this chapter to draw the links between those methods (the AIM being often based on the gradient projection method), and especially to summarily explicit the COM frequently used in robotics reactive control (Annex C).

The contributions exposed in this chapter are based on this combination: the Constraint Compliant Control (CCC) is a complete algorithm that relies on passive avoidance to comply with strict tasks while resorting to active avoidance to maintain a well conditioned problem and reach the objective tasks. The displaced point principle enlarges the scope of the CCC and opens a way to solve problems involving inequality tasks in a single iteration by means of a (reduced) optimality loss.

The particular case of *Alternative Safe Behaviors* is not treated in this chapter; if an ASB is required at a given time step, it is detected by a prediction consecutive to the problem resolution (cf. section 2.4.5).

As a vocabulary remark, a difference is made between *time steps*, which denote the control instants k, $k+1$, etc. and *iterations*, which denote the computation of internal algorithm loops.

3.1 State-of-the-art

The control problem formulation proposed in chapter 2 involves constraints (inequalities) expressed as conditions to be satisfied at all costs. However, as mentioned above, this is not the only way the control problem is addressed in the literature. To differentiate from the previous framework where the operational inputs where described with *objectives* and *constraints*, the generic term *task* is used. The present state-of-the-art reviews the most usual ways to solve the control problem according to its formulation. A special attention is paid to draw the links between direct AIM (the most frequently used in robotics) and COM. A quick presentation of COM is proposed in appendix C. It is limited to methods actually used in Inverse Velocity Kinematics (IVK) problems, *i.e.* the one exploiting the particular structure (especially the fact that the cost function is generally quadratic) of the problem.

Section 3.1.1 exposes various ways to express the control problem. Sections 3.1.2 and 3.1.3 detail single and multiple hierarchical levels problems, solved with AIM and COM. Sections 3.1.4 and 3.1.5 introduce inequality tasks in the control problem; in this case, only COMs remain.

3.1.1 Problem specifications

The problems addressed in control may vary according to the need of the user. Even if high level control loops such as the ones of planning or supervision are out of the scope of the present work, the way the user specifies his need has an impact on the problem formulation. Among the operational inputs, the simplest way to specify tasks is to impose one or several targets with relative importance (section 3.1.2). However, this cannot be sufficient for demanding tasks: the framework exposed in chapter 2 considers that independently of any specification type, the control law should be able to satisfy tasks (constraints in this case) without influence (strict priority) of other potential tasks. The problem is said to be formulated through strict hierarchical levels (section 3.1.3). The management of these tasks into the controller scope (appearance, transitions, removal) is not addressed in this document. The interested reader can refer to the work of Mansard *et al.* (2007), Keith *et al.* (2009), Padois *et al.* (2007) and Salini *et al.* (2011). Finally, the tasks are almost always specified in terms of equalities, probably because the usual solving methods (AIM) are likely to solve these kinds of problems. But basically, there are many kinds of tasks that are relevant to be specified as inequalities (section 3.1.4).

To illustrate the various methods referenced in the following literature review, the case of a 4-DOFs planar manipulator in charge of observing a star-shaped object in a cluttered environment is proposed (Fig. 3.1). The presence of the text "you should read this text toward the star" is here to illustrate that the robot should track a particular trajectory to reach the star (a letter is red once the camera gets over it) and that any tracking error is a drawback as the message cannot be red completely. The velocity kinematic model of the camera motion is

$$\dot{\boldsymbol{X}}_O = J_O \dot{\boldsymbol{q}}. \tag{3.1}$$

$\dot{\boldsymbol{X}}_O$ is the camera velocity in the Cartesian space (size($\dot{\boldsymbol{X}}$)=3, 2 linear velocity and 1 angular velocity), J_O is the Jacobian of the camera (size(J_O)=(3,4)) and $\dot{\boldsymbol{q}}$ is the vector of joint velocity (size($\dot{\boldsymbol{q}}$)=4). The manipulator is said *redundant* with respect to the

camera positioning task as there are more DOFs (4) than required (3) to move the camera at a given position in the robot workspace.

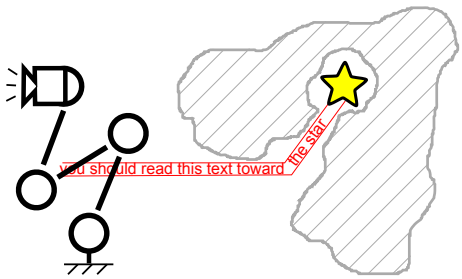

Figure 3.1: 4 DOFs planar manipulator in a cluttered environment.

3.1.2 Single hierarchical level

In this section, all the tasks assigned to the system are expressed with relative but not absolute importance (no strict priority).

3.1.2.1 Problem formulation

The problem is expressed through an equality expression:

$$\text{Find } \dot{\boldsymbol{q}}(k+1) \text{ such that } \dot{\boldsymbol{X}}_{des,O}(k+1) - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1) = \boldsymbol{0} \tag{3.2}$$

where $\dot{\boldsymbol{X}}_{des,O}(k+1)$ is a concatenation of the desired operational inputs associated to Jacobian $J_O(\boldsymbol{q})(k)$ evaluated at current time step k and $\dot{\boldsymbol{q}}(k+1)$ is the vector of joint velocities to find at time step k so that $\dot{\boldsymbol{X}}_{des,O}(k+1)$ is performed at time step $k+1$.

The convex optimization version of this problem is an unconstrained quadratic optimization problem

$$\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des,O}(k+1) - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)||. \tag{3.3}$$

This problem expression means that the aim of the resolution method is to find a control solution $\dot{\boldsymbol{q}}(k+1)$ so that $J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)$ is as close as possible to the desired operational velocity $\dot{\boldsymbol{X}}_{des,O}(k+1)$. This way of considering the problem and the associated resolution methods are exposed in appendix C.1. The particular case of inequalities in a single priority level (rarely addressed) is described in 3.1.5.

3.1.2.2 Resolution methods

For clarification, the dependence of the Jacobians to $\boldsymbol{q}(k)$ and the dependences to the time steps k and $k+1$ are no longer mentioned. When not explicitly mentioned, the joint velocity $\dot{\boldsymbol{q}}$ denotes the control vector $\dot{\boldsymbol{q}}(k+1)$, and the other values are the e-state variables expressed at time step k. The range of a linear application represented by matrix A is denoted $Rg(A)$; the kernel of a linear application represented by matrix A is denoted $Ker(A)$.

The formulation of Eq. (3.2) enlightens the problem of the Jacobian inversion. The general (potentially approximated) solution proposed by Liégeois (1977) is

$$\dot{\boldsymbol{q}} = J_O^{\#}\dot{\boldsymbol{X}}_{des,O} + P_{J_O}\boldsymbol{z} \tag{3.4}$$

where $J_O^{\#}$ is a weighted pseudoinverse of J_O, P_{J_O} is a projector on $Ker(J_O)$ and $\boldsymbol{z}$ an arbitrary vector of $\mathbb{R}^n$. $J_O^{\#}\dot{\boldsymbol{X}}_{des,O}$ is the particular solution of Eq. (3.2) while $P_{J_O}\boldsymbol{z}$ is its homogeneous solution (see appendix B for more information on the inversion of the linear problem of Eq. (3.2)). Considering a single hierarchical level means that only the term $J_O^{\#}\dot{\boldsymbol{X}}_{des,O}$ is used ($\boldsymbol{z} = 0$).

In the case of the 4 DOFs manipulator, an example of task is to reach the star

$$\dot{\boldsymbol{X}}_{des,O} = k_1 \frac{\boldsymbol{X}_{des,O} - \boldsymbol{X}_O}{\delta t} \tag{3.5}$$

where k_1 is an error gain, $\boldsymbol{X}_{des,O}$ is the desired end-effector position and orientation at the current time step, $\boldsymbol{X}_O$ is the current end-effector position and orientation and δt is the time step. However, this task would lead to collisions as the environment is not taken into account (Fig. 3.2).

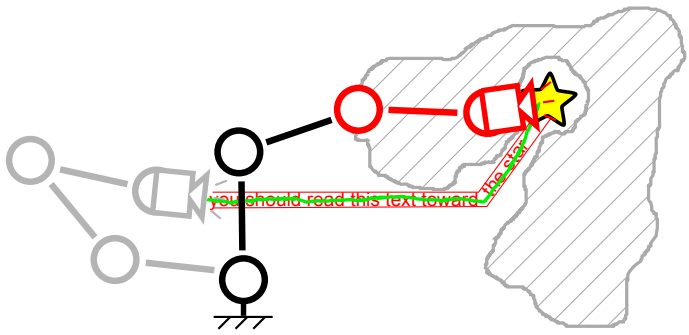

Figure 3.2: The robot tries to go straight to the star without considering the environment.

To avoid these obvious collisions, Khatib (1986) considers potential fields (attractive for the robot target, repulsive for the obstacles) to generate virtual forces on the robot (Fig. 3.3). However, this approach generates numerous operational inputs on the manipulator, making the control law unable to satisfy all of them simultaneously. As a consequence, it is not possible to ensure the permanent satisfaction of all tasks, and envelopes around the obstacles are needed. As a result, the compromise between repulsive and attractive fields does not yield optimal results.

Other single hierarchical level approaches involves the augmented matrix approach (Bailleul *et al.* 1985) which proposes to fit the tasks of a redundant system so that the Jacobian remains a square invertible matrix. To this end, additional tasks are inserted: matrix singularities avoidance and/or tasks making closed operational trajectories generate closed joint trajectories, which is of interest in industrial applications for example. In the same scope, Seraji proposed in 1989 the configuration control approach (Seraji *et al.* 1989), improved later (Seraji *et al.* 1990) thanks to a singularity-robust task-prioritized reformulation. More recently, Shen *et al.* (2007) solved the control problem by minimizing a weighted sum of task errors. Similarly to the approach of Khatib (1986), although these methods meet quite well the single hierarchical level specifications, they suffer from the limits inherent to these specifications: they consider a limited number of tasks with the same priority. However, they do not meet for example the needs expressed in chapter 2 (satisfy the constraints, and, if possible, the objectives).

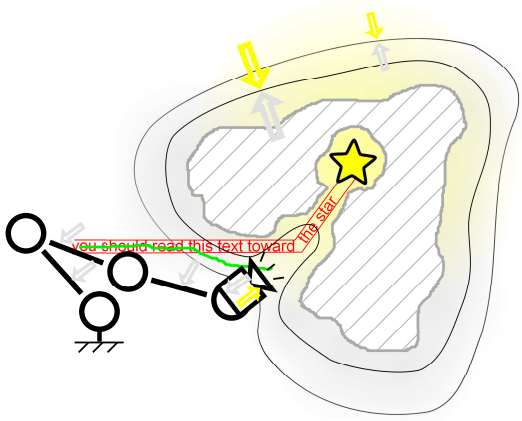

Figure 3.3: The robot is submitted to repulsive (obstacles) and attractive (star) forces. The resulting compromise is not theoretically safe and not optimal.

3.1.3 Multiple hierarchical levels

This section deals with multiple hierarchical levels specifications for which strict priority levels are defined: influences of the performance of low priority tasks on the performance of higher priority tasks are forbidden.

3.1.3.1 Problem formulation

The multiple hierarchical levels control is the most largely used control problem formulation. In this case, tasks are strictly prioritized: any performance for a given level can be carried out as long as it does not cause a deterioration of the tasks at higher levels. A representative expression of such a problem can be done through a sequence of Least Square problems with Equality constraints (LSE)

$$\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{(i+1)} - J_{(i+1)}\dot{\boldsymbol{q}}|| \tag{3.6}$$

$$\text{subject to} \quad J_{(i)}\dot{\boldsymbol{q}} - \dot{\boldsymbol{X}}_{(i)} = 0 \tag{3.7}$$

where $J_{(i)}$ is the Jacobian of tasks of the i^{th} hierarchical level (the higher the index, the lower the priority) and $\dot{\boldsymbol{X}}_{(i)}$ the associated operational input. $||\dot{\boldsymbol{X}}_{(i+1)} - J_{(i+1)}\dot{\boldsymbol{q}}||$ is the Cost Function (CF), and $J_{(i)}\dot{\boldsymbol{q}} - \dot{\boldsymbol{X}}_{(i)} = 0$ are the Equality Constraints (EC). The resolution of this LSE with COM algorithms is described in appendix C.2. Its resolution with AIM is exposed hereafter.

3.1.3.2 Resolution through gradient projection method

Based on the expression of Liégeois (cf. Eq. (3.4))

$$\dot{\boldsymbol{q}} = J_O^{\#}\dot{\boldsymbol{X}}_{des,O} + P_{J_O}\boldsymbol{z}, \tag{3.8}$$

an exploration of the null space of the main task $(J_O, \dot{\boldsymbol{X}}_{des,O})$ is possible through the $\boldsymbol{z}$ term projected on the Jacobian kernel by P_{J_O}. This method is often called the Gradient Projection Method in the literature as the potential (magnitude to be decreased) from

which $\boldsymbol{z}$ is derived (gradient) is projected on the kernel of the high priority task Jacobian J_O. This approach is the application of the COM Gradient Projection Method in the case of a quadratic problem. An extension of Eq. (3.4) enables to take an arbitrary number of tasks into account (Siciliano *et al.* 1991)

$$\dot{\boldsymbol{q}}_{(i)} = \dot{\boldsymbol{q}}_{(i-1)} + \bar{J}^{\#}_{(i)}(\dot{\boldsymbol{X}}_{(i)} - J_{(i)}\dot{\boldsymbol{q}}_{(i-1)}),\ \dot{\boldsymbol{q}}_{(1)} = J^{+}_{(1)}\dot{\boldsymbol{X}}_{(1)} \tag{3.9}$$

with

$$\bar{J}_{(i)} = J_{(i)}P_{(i-1)},\quad P_{(i)} = I - J^{\#}_{(i)}J_{(i)},\quad J_{(i)} = \begin{bmatrix} J_{(1)} \\ J_{(2)} \\ \vdots \\ J_{(i)} \end{bmatrix}. \tag{3.10}$$

A large part of the literature dedicated to constraints avoidance techniques (*e.g.* obstacles or joint position limits) resorts to strictly prioritized problems. Active avoidance techniques, such as the one introduced by Maciejewski *et al.* (1985), tends to get the robot away from areas by imposing velocities on the segments at the lowest hierarchical level. The proposed control law is then

$$\dot{\boldsymbol{q}} = J^{\#}_{O}\dot{\boldsymbol{X}}_{des,O} + (J_{obst}P_{J_O})^{\#}(\dot{\boldsymbol{X}}_{des,obst} - J_{obst}J^{\#}_{O}\dot{\boldsymbol{X}}_{des,O}) \tag{3.11}$$

where J_{obst} is the concatenation of the Jacobians of closest obstacles points to the robot (1 by segment maximum) - referred as $J_{A,B}$ in section 2.4.2. To increase safety, the hierarchy is inverted ($(J_O, \dot{\boldsymbol{X}}_{des,O}) \leftrightarrow (J_{obst}, \dot{\boldsymbol{X}}_{des,obst})$), avoidance at the highest level). The method is generalized by Sentis *et al.* (2005) through a whole body motion framework. This approach imposes to use activation thresholds in order to define areas where the avoidance motions are active. Else, the robot would be permanently avoiding obstacles without considering the trajectory tracking. The comparative behaviors of the 4 DOFs manipulator submitted to the control law of Maciejewski and Sentis are represented on Fig. 3.4.

The principle of strict priority between hierarchical levels has been soften by Mansard *et al.* (2009b) who introduced *directional redundancy* to enable favorable influences of low hierarchical levels on high hierarchical levels. Mansard *et al.* (2009a) addressed the problems of discontinuity induced by Jacobian rank changes (the continuity being not explicitly taken into account as a task itself).

As partially shown on Fig. 3.4 these techniques suffer from severe drawbacks:

- They overconstrain the robot, by imposing arbitrary inputs on it; when too many conditions are imposed (more than 1 by DOF), the solution found at best minimizes the error toward the desired behavior. As the expected behavior is not feasible, the robot is in a control loss (does not strictly behave as specified by the user), which is critical in presence of constraints;

- To overcome the lack of safety, obstacles and constraints are surrounded by envelopes of arbitrary width, which decreases the working space of the robot without strictly ensuring safety;

- They generate oscillations at the activation threshold. For example, if a task tends to get the robot closer to obstacles: 1/ the robot is inside the obstacle activation area → active avoidance; 2/ the robot gets outside the obstacle activation

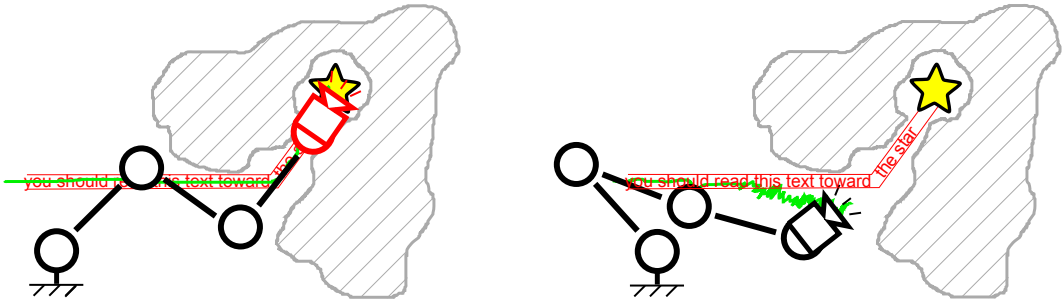

Figure 3.4: Comparison between obstacle avoidance at the second (Maciejewski *et al.* 1985) and the first level (Sentis *et al.* 2005). **Left**: the robot tracks perfectly the trajectory and the configuration obtained maximizes the distance to the obstacles while maintaining the camera on the star. However, as it is not possible to reach the star without obstacle avoidance, a collision occurs. **Right**: the robot cannot properly tracks the trajectory as it avoids the obstacles in first priority. The avoidance methods generate oscillations on the camera at the avoidance activation threshold; the camera alternates between 2 states as the trajectory and the obstacles have opposite effects at different priority levels. Despite its cautious behavior this method is not safe, as exposed in Fig. 3.5.

area → no obstacle avoidance, motion toward the obstacles; 3/ back to 1. This phenomenon is limited if the active avoidance is at a low hierarchical level: as the superior tasks are not impacted by the oscillations, there is no particular reason for the configuration to be maintained in the neighborhood of activation thresholds. However it increases the risks of safety violations;

- The addition of potentials (as in the case of competition between joint boundaries and obstacles for example, see Fig. 3.5) may lead to infinite inputs and direct violations.

3.1.3.3 Resolution through Jacobian inversion weighting methods

Another way to explore the kernel of the high level tasks is to modify the norms (in operational and joint spaces) being minimized by the pseudoinverse (for calculus details about weighting see appendix B.3.1 and refer to Doty *et al.* (1993) and Ben Israel *et al.* (2003)). Briefly, in most of the cases (J_O is full rank in line) the pseudo-inversion is influenced by weights attributed to the joints: the operational motion being satisfied, the repartition of this motion between the joint is done accordingly to the weighted norm minimization. The avoidance in this case is not based on an *active* but on a *passive* approach: the avoidance does not generate a motion, it rather slows or prevent motions. The weight attributed to a joint can get so important that the control gets progressively unable to resort to it. This possibility is used for joint boundaries avoidance by Chan *et al.* (1995). Huo *et al.* (2011) proposed a self adapting weighting coefficient algorithm that increases the performances of joint limits and singularity avoidance. This approach was recently generalized by Xiang *et al.* (2010) who, by introducing virtual joints, enable to handle an arbitrary number of constraints by avoiding virtual joints boundaries. To our knowledge, this approach has never been applied to obstacles avoidance.

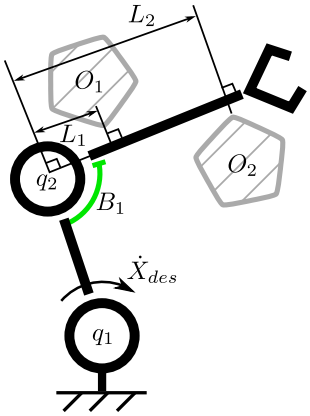

Figure 3.5: Example of competition between obstacles and joint boundary avoidance tasks. In this example, the controller has 2 tasks strictly prioritized: 1/ Moving q_1 according to $\dot{X}_{des,O}$; 2/ Avoiding the two obstacles and the joint boundary. The second task involves 3 avoidances that are in competition to prevent the violation of their associated constraints. **Between the two obstacles**: each obstacle generates avoidance velocities on the closest point of the robot, which magnitude depends on the obstacle proximity. Given the problem configuration, the second segment of the robot converges toward a position which proximities to O_1 and O_2 depends on the distance L_1 and L_2. This becomes a problem when q_1 continues its motion as L_1 tends to 0, which should provoke an infinite avoidance magnitude from obstacle O_1. This is likely to lead to a collision with O_2 in discrete time control. **Competition with the joint boundary**: the joint boundary avoidance is computed differently from the obstacles avoidance as there is no length L_3 associated to it. So, a normalization should be done. Anyway, the behavior induced by the avoidance of O_1 in its competition with O_2 is likely to violate joint boundary B_1 as well. This behavior is not acceptable as it seems that, until the second joint reaches O_2, non violating solutions exist.

As the gradient projection method, the weighting methods exploit the null space (redundancy) of a task. If both methods are used to explore the same null space at the same hierarchical level, the gradient projection task is effectively carried out at a superior sub-level than the weighting method. Actually, the weighting influences the Jacobian inversion while the projection on the Jacobian kernel is an additional term that takes the inversion into account to exploit directly the redundancy, thus overlapping any operation carried out on the inversion.

Classical weighting techniques use positive definite matrix to define the weights (Ben Israel *et al.* 2003). Using semi-positive definite matrix may violate the strict hierarchy by having influences on top priority tasks, which can punctually be useful (typically for constraints imperative satisfaction). This technique is exploited by Xiang *et al.* (2010). However, discontinuities in the solutions occur when the matrix passes from positive to semi-positive definition.

A more disturbing drawback is that weighting cannot deal with tasks expressed as inequalities: the weighting coefficient are fixed before the model inversion, *i.e.* without knowledge about the direction of the solution along the tasks. For example, if a joint is close to its boundary, its weight is high, so it has difficulties to get closer as well as further from the boundary. The only way to remedy to this drawback is to resort to

iterations (*e.g.* in Xiang *et al.* (2010)) to incrementally fix the weights.

3.1.4 Use of inequality tasks in IVK problems

As described in 3.1.3, tasks associated to limits (inequalities by essence) are most of the time considered through equalities, probably because of the AIM resolution structure. The introduction of inequalities in the problem formulation has been for a long time addressed only by COM. Conversely to equality objective problems, any problem including inequalities requires iterative algorithms to be solved (excepted for the method proposed by Mansard *et al.* (2009a), which relies on multiple Jacobian inversions and is very time-consuming). The most efficient way to solve a problem involving inequalities is to consider them as equalities. This is the principle of active sets methods for example (see section 3.1.4.3), which key to solve an inequality constrained problem is to find the appropriate equality constrained problem to be considered.

3.1.4.1 Problem formulation

There can be many ways of specifying a control problem involving inequalities. However, most of them can be gathered in 2 categories:

- The inequality is an objective (*e.g.* an area to reach in the operational space): this problem has been neglected for a long time, the only contribution found in this field is the one of Kanoun *et al.* (2009), described in section 3.1.5.
- The inequality is a constraint (*e.g.* a forbidden area in the operational space): this problem is far more common as it describes well the physical limits of robotic systems, it is formulated through Least Square problem with Inequality constraints (LSI).

A LSI can be expressed ad

$$\min_{\dot{\boldsymbol{q}} \in \mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des,O} - J_O \dot{\boldsymbol{q}}|| \tag{3.12}$$

$$\text{subject to} \quad J_C \dot{\boldsymbol{q}} - \boldsymbol{b} \leq 0. \tag{3.13}$$

In this problem, the tasks are divided into objectives (J_O, $\dot{\boldsymbol{X}}_{des,O}$) and constraints ($J_C$, $\boldsymbol{b}$). The problem involves a single objective (or a set of objectives at the same hierarchical level) and possibly inequality constraints. Many techniques can be used to solve such a problem with COM. For example, interior points methods turns this problem into a sequence of unconstrained ones by adding the constraints to the cost function with an increasing penalizing weight. Other methods frequently used with quadratic programs are active-sets methods. These methods are based on a simple observation: the solution of an LSI can be obtained by the resolution of a LSE constrained by a subset of the inequality constraints taken as equalities. The key is to determine which inequalities are active (= the optimal solution satisfies it as an equality). Both methods are regularly used in Model Predictive Control (MPC) (Bartlett *et al.* 2000); for example in the case of IVK problems, active sets methods are the most widely used. The reason is that usual IVK problems are small (with respect to MPC ones) and active constraints are quickly identified.

As an example, the primal active sets method is briefly exposed hereafter (section 3.1.4.3. For a more detailed description, refer to appendix C.3 and Nocedal *et al.* (2000)). As an introduction, the work of Baerlocher *et al.* (2004) is described (section 3.1.4.2).

3.1.4.2 Joint clamping on a humanoid

Baerlocher *et al.* (2004) propose an algorithm to satisfy joint boundaries constraints on a humanoid robot. At each time step, a whole model inversion is done; if the obtained motion violates some joints position limits, then these joints are clamped to their boundaries. The resulting operational motion is taken into account and a new model inversion is done, without considering the previously clamped joints. Again, if the motion violates other joints position limits, these joint are clamped, etc. At the end of the computation, the final joint motion is sent to the actuators and all the joints are unclamped for next time step. This algorithm is presented on Fig. 3.6.

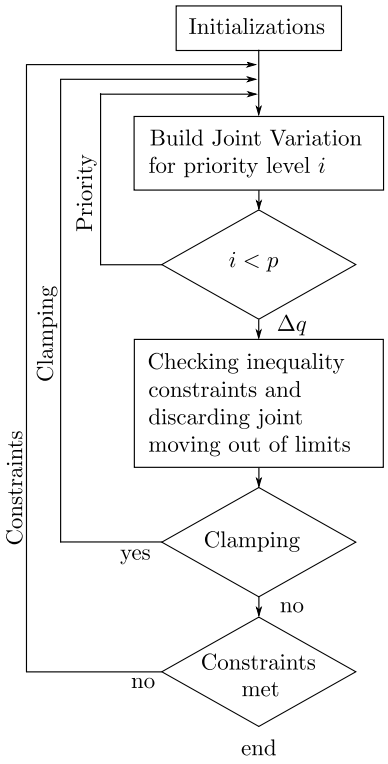

Figure 3.6: Prioritized Inverse Kinematics algorithm (Baerlocher *et al.* 2004): once the prioritized joint variation is obtained as the output of the inner loop (priority loop), the updated configuration is checked for joint limit violation. Any detected subset of violating variations leads to the introduction of temporary equality constraints that clamp the corresponding joints on their respective limit. The prioritized solution is re-evaluated with this updated context as long as additional limit violation is detected (clamping loop).

This approach finds incrementally the set of active constraints by clamping the violated joint position limits in a try/check/clamp iterative structure. The convergence of this algorithm is illustrated on Fig. 3.7. The projectors on the tasks kernel are built

iteratively to increase the algorithm efficiency. In more recent works (Raunhardt *et al.* 2007, Peinado *et al.* 2009), the problems of discontinuity at the velocity level induced by clamping is circumvented and obstacle avoidance management is included in the framework).

However, despite optimized computation methods (PIK), these approaches lacks of efficiency: as there is no indication whether an active constraint should be relaxed or not, the active set begins empty at each time step and grows progressively, which is time-consuming in very constrained cases. Moreover, for the same reason, in a few case this method is non optimal (Fig. 3.8).

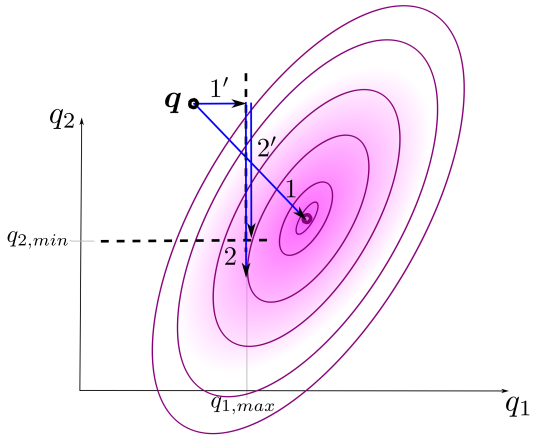

Figure 3.7: Representation of the clamping method convergence (Baerlocher *et al.* 2004) in the joint space. $\boldsymbol{q}$ is the current configuration. The point to reach is the centre of the equipotential ellipses. Solution 1 reaches the minimum of the unconstrained cost function but exceeds the first joint boundary. Joint 1 is then clamped (1') and a new solution is computed. Solution 2 is then computed with joint 1 clamped; the solution 2 exceeds the boundary of joint 2. Joint 2 is thus clamped (2'). The final motion is the concatenation of 1' and 2'; it is the optimal solution for the given problem.

3.1.4.3 Primal active sets method

The active sets methods enables to get information on the constraints that should be activated and/or relaxed along the iterations in order to find the adequate LSE yielding the optimal value. A short description of the primal active set method is given hereafter. The detailed algorithm and an example are given in appendix C.3. A rigorous and complete approach is given in Nocedal *et al.* (2000).

The primal active-set algorithm begins by a *phase I* problem, *i.e.* by finding a feasible point (this problem is assumed to be solvable). Once found, the algorithm addresses *phase II*, *i.e.* the optimal resolution of a constrained problem from an admissible initial solution. Similarly to the clamping method, this algorithm looks for the set of constraints that are active at the optimal point. Given a subset J_c of all the constraints J_C and an associated admissible point, an optimal solution $\dot{\boldsymbol{q}}$ is found (LSE problem).

- If one or more constraints of J_C are violated by $\dot{\boldsymbol{q}}$, then the solution is scaled to fit the most constraining constraints and this constraint is added to J_c; a new LSE problem is to be solved.

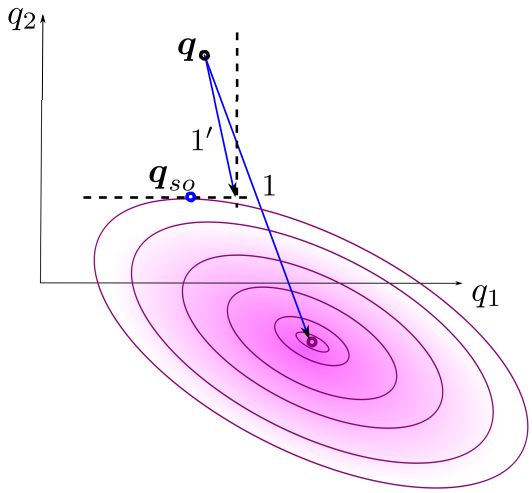

Figure 3.8: Example of non optimality of the clamping method: 2 joint boundaries are violated simultaneously, so solution 1 is clamped on both boundaries (solution 1'). However, the optimal solution is at the blue point (tangency with equipotential curves).

- Else, $\dot{\boldsymbol{q}}$ is submitted to a Karush Kuhn Tucker (KKT) optimality condition (see appendix C.3, Eq. (C.25))
  - in case of success, the solution is optimal.
  - else, a constraint must be removed from J_c; a new LSE problem is to be solved.

The important step (that could be introduced with much interest in the clamping method) is the submission to the optimality condition; it would enable the clamping method algorithm to initialize the subset of constraints to a non empty subset, which should be very profitable in very constrained situations.

As a remark, the optimality conditions impose the inversion of a linear system, which requires that the set of constraints is linearly independent. As a consequence, a preliminary conditioning work is to be done at the beginning of each iteration.

The computational efficiency being critical for such methods, Baerlocher *et al.* (2004) propose incremental computation for projectors; Similarly, Nocedal *et al.* (2000) present factorization updates. A recent contribution of Escande *et al.* (2010) introduce the use of the Complement Orthogonal Decomposition (COD) based inverse (instead of the widely used Singular Value Decomposition) in an active set based algorithm which is such that complexity is less sensitive to the number of constraints. The computation times obtained are impressive and sufficiently low to consider the use of such an algorithm in real time control of humanoids (see Fig. 3.9). However, as a drawback inherent to such methods, the research for the optimal active constraints can be costly; the peaks in Fig. 3.9 multiplies the computation times by 5.

The behavior obtained with such methods is generally satisfying; they enable an appropriate management of priorities and present a correct behavior (no avoidance oscillations as the method is passive, no motion is generated). However, these methods often *stick* to the constraints as there is no avoidance motion generated by the control. Moreover, they are not adapted to avoid concave obstacles (Kanehiro *et al.* 2008). As a result, they have difficulties to converge in cluttered environments.

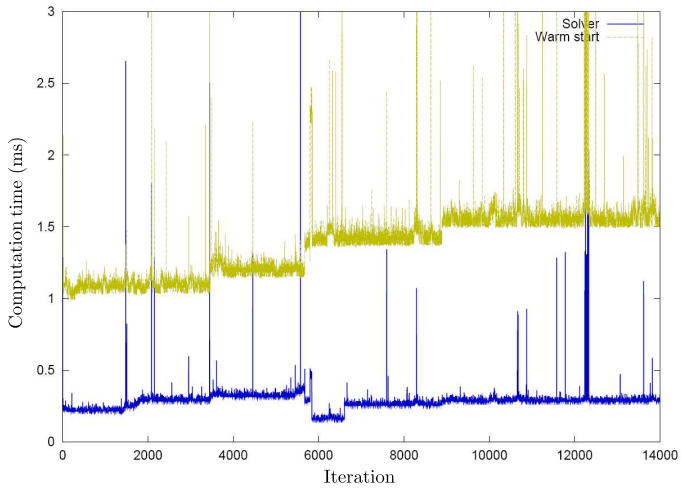

Figure 3.9: Figure taken from Escande *et al.* (2010). Computation times obtained by Escande *et al.* (2010) for a 36 DOF humanoid submitted to more than 50 constraints. The times of the yellow curve and the blue curve are to be added.

3.1.5 Inequalities as objectives

Considering inequalities as objectives for a robot is rarely addressed; however it can be the case for example if the robot should be maintained in an area without specifically giving him a target point. In this case, the problem can be expressed by:

$$\min_{\dot{\boldsymbol{q}} \in \mathbb{R}^n} ||\sup_v(J_i\dot{\boldsymbol{q}} - \boldsymbol{b}, \boldsymbol{0})|| \tag{3.14}$$

where $\sup_v$ is the component-by-component function that returns a vector containing the highest values of its two arguments. For example, Fig. 3.10 shows a case where inequalities and equalities (each can be expressed by 2 inequalities) are taken into account at the same hierarchical level.

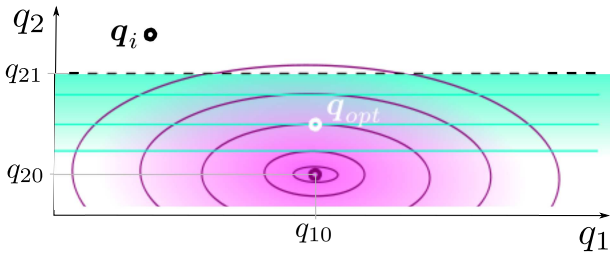

Figure 3.10: Problem considering 2 equalities (that can be expressed by 4 inequalities: $q_{1,des} \leq q_{10}$, $q_{1,des} \geq q_{10}$, $q_{2,des} \leq q_{20}$, $q_{2,des} \geq q_{20}$) and 1 inequality ($q_{2,des} \geq q_{21}$) objectives at the same hierarchical level. $\boldsymbol{q}_i$ is the initial point. The optimal point $\boldsymbol{q}_{opt}$ is equidistant from the inequality boundary and the equality desired configuration.

The problem is not addressed in this form in optimization as it is not derivable. This problem can be re-expressed by the following problem using slack variables (Kanoun *et*

al. 2009)

$$\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n, \boldsymbol{w}\in\mathbb{R}^p} \quad ||w||^2 \tag{3.15}$$

$$\text{subject to } J_C\dot{\boldsymbol{q}} - \boldsymbol{b} \leq \boldsymbol{w}. \tag{3.16}$$

In this case, the vector on which to carry out the optimization is $[\dot{\boldsymbol{q}}^T, \boldsymbol{w}^T]^T$. This problem is then a Least Square Problem submitted to Inequality constraints (LSI), which can be addressed as described above.

$$\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n, \boldsymbol{w}\in\mathbb{R}^p} \quad ||w||^2 \tag{3.17}$$

$$\text{subject to } \begin{bmatrix} J_C & -I \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^T & \boldsymbol{w}^T \end{bmatrix}^T - \boldsymbol{b} \leq 0. \tag{3.18}$$

3.1.6 Summary

The control laws proposed in this literature review are all fitted to their associated problem. The most generic problem formulation (hierarchical with inequalities) is the only one able to deal with the framework exposed in chapter 2. It is addressed by iterative algorithms such as the clamping methods or the LSI COM. However, it seems that these algorithms can reach high computation times; the clamping algorithm is limited to a certain types of constraints (joint limits); active set algorithms are not always adapted to complex robotics environment (no constraints avoidance).

3.2 Constraint Compliant Control

The so-called Constraint Compliant Control (CCC) is a control problem resolution algorithm developed for the particular context of an evolutionary design process that resorts to trajectory trackings in a cluttered environment to evaluate successively the kinematic performances of a huge number of robotic systems (part 4.2). In this context, the CCC should be applicable to any kind of serial robot (redundant or not with respect to a 3D trajectory tracking), with the prerequisite that no collision should occur with the environment to have a representative and relevant evaluation. As the environment is cluttered, most control laws described previously are inadequate, which could penalize the robots arbitrarily.

In a first section, the context and the assumptions are briefly summed up. The second section is structured on the basis of the desired features of the CCC: safety with respect to the considered constraints, ability to address various constraints, acceptable behavior. Then, the third section focuses on the algorithm and its implementation details. A conclusion sums up the CCC schemes and its expected performances.

The CCC as described below is limited to problems involving constraints which control admissibility space ($\mathscr{C}(\boldsymbol{\sigma})$, see 2.3.1) contains the null vector (i.e. instantaneously stopping the robot is admissible with respect to the constraints, in other words the joint acceleration limits are not taken into account). This scope is enlarged in section 3.3 thanks to the displaced configuration technique.

3.2.1 Context and considered constraints

The control problem has been largely discussed in previous sections (2.1.3, 2.1.4). It is summarized through the following assumptions:

- The control problem is considered at the velocity kinematic level: at each time step, the aim is to find joint velocities $\dot{\boldsymbol{q}}(k+1)$ from the operational inputs $\dot{\boldsymbol{X}}_{des,O}(k+1)$;
- The models of constraints represent reality with an infinite precision;
- The sensors measure reality perfectly;
- The joint inputs are carried out by the actuators perfectly and in a single time step;

The considered constraints in the context of the design problem are kinematic (simulation):

- Joint position limits;
- Joint velocity limits, to satisfy the assumptions of small motions between time steps (the Jacobian is a local model);
- Obstacles avoidance.

According to section 2.4.2, the 3 constraints functions associated to these 3 constraints are compatible: actually, the null motion solution is always admissible. The CCC exploits this particularity.

3.2.2 Safety and passive avoidance

The notion of safety, as defined in chapter 2, requires the control problem resolution algorithm to be able to satisfy an arbitrary set of compatible constraints. As each equality imposed to a controller requires one DOF in the general case, it is important to maintain the number of equality constraints as low as possible. In particular, it seems hazardous to try to satisfy inequality constraints by introducing equality constraints corresponding to avoidance terms. In that approach, the passive avoidance is a principle according to which the robot should not move to avoid static constraints (constraints which expression does not change when the robot does not move).

3.2.2.1 Passive avoidance

The principle of passive avoidance is that a static constraint should not impose a motion to a moving system, but rather an absence of motion. This principle is able to deal with an arbitrary number of constraints, as all of them impose the same condition (absence of motion). The Jacobian of these constraints being concatenated in a matrix J_c, the application of this principle yields

$$\begin{aligned} \dot{\boldsymbol{q}}(k+1) &= J_c(\boldsymbol{q}(k)) \;\; \mathbf{0} \\ &= \mathbf{0}. \end{aligned} \tag{3.19}$$

This term is always null, but the choice of matrix J_c is essential: the exploration of its kernel prevents the robot to move along the constraints directions.

To know which constraint should be in J_c or not, an iterative method (detailed in section 3.2.4) is used.

3.2.2.2 Admissibility of the Null motion

In the retained context, the null motion is always admissible, i.e. the control problem solution $\dot{\boldsymbol{q}}(k+1) = \mathbf{0}$ satisfies the constraints whatever the situation. As a consequence, Eq. (3.19) can be directly the first hierarchy level of the control law: in the most constraining case, J_c covers all the system DOFs (robot blocked), and the resulting motion is always null (and safe).

3.2.2.3 Comparison with other approaches

This no-motion high hierarchy level is an extension of the clamping algorithm (Baerlocher *et al.* 2004), any violating motion being somehow clamped to its current position. In the original clamping algorithm, the equivalent of J_c is limited to a diagonal matrix composed of 0 / 1 terms, representing the non-active / active joint boundaries; here, it can include any kind of static constraints.

In a convex optimization point of view, this no-motion high hierarchy level is the equivalent of the constraints subject to which the problem is solved. The other tasks are expressed in terms of cost functions.

3.2.3 Correct behavior and active avoidance

This section introduces the other hierarchy levels of the CCC control law.

3.2.3.1 Objectives

In accordance with the specifications of chapter 2, the trajectory tracking comes at a lower hierarchical level than constraints. It is expressed as a set of equalities concatenated in a Jacobian matrix J_O.

$$\begin{aligned}\dot{\boldsymbol{q}} &= J_c^+\mathbf{0} + (J_O P_{J_c})^+(\dot{\boldsymbol{X}}_{des,O} - J_O J_c^+\mathbf{0}) \\ &= (J_O P_{J_c})^+\dot{\boldsymbol{X}}_{des,O}. \end{aligned} \tag{3.20}$$

This set of equality tasks can be expressed through several hierarchical levels, but always below the passive avoidance task (J_c, $\mathbf{0}$).

3.2.3.2 Active Avoidance

Conversely to passive avoidance, active avoidance methods impose motions on the system. Active avoidance has a lot of interests with respect to passive avoidance, in particular it increases the avoidance robustness:

- with respect to perturbations (approximate distance measures for example) in feasible problems as it maintains the robot away from its constraints;
- with respect to local minima in a more global scope.

However, as seen in section 3.1.3.2, active avoidance has many drawbacks (briefly summarized here):

- It overconstrains the robot, by imposing numerous arbitrary inputs on it;

- It reduces the working space of the robot by envelopes around obstacles and constraints in general;
- It generates oscillations at the activation thresholds, especially for high priority avoidance tasks;
- The addition of potentials may lead to infinite inputs.

3.2.3.3 CCC law

In order to benefit from both passive and active avoidance, at each time step, all the problem constraints are concatenated in the matrix J_C, which is divided into constraints avoided passively J_c (chosen among the most critical constraints) and actively $J_{\bar{c}}$. An appropriate repartition of constraints between passive and active avoidance is not easy to find *a priori*[1] and an iterative scheme is required. The CCC law is then

$$\dot{\boldsymbol{q}} = \underbrace{(J_O P_{J_c})^+ \dot{\boldsymbol{X}}_{des,O}}_{\text{Objective tracking}} + \underbrace{\left(J_{\bar{c}} \; P_{\begin{bmatrix} J_c \\ J_O \end{bmatrix}}\right)^+ (\dot{\boldsymbol{X}}_{des\bar{c}} - J_{\bar{c}}(J_O P_{J_c})^+ \dot{\boldsymbol{X}}_{des,O})}_{\text{Active avoidance}} \tag{3.21}$$

where $P_{\begin{bmatrix} J_c \\ J_O \end{bmatrix}}$ is the projector on the concatenation of J_c and J_O, and $J_{\bar{c}}$ is the complement of the lines of J_c in J_C. In a practical aspect, $J_{\bar{c}}$ can be replaced by J_C in Eq. (3.21) without any consequence on the result. The active avoidance term tends to move the manipulator away from the constraints as long as the objective is not impacted.

To avoid the drawbacks related to infinite terms, the potential from which the avoidance term $\dot{\boldsymbol{X}}_c$ is differentiated is limited to a predefined value. Actually, this can be done safely as passive avoidance enforces the constraints satisfaction anyway. Finally, there are no oscillations on the functional part as the active avoidance term is lower than the objective related terms in the hierarchy. For each constraint, an active avoidance threshold δ is defined as a distance under which the avoidance is switched on.

3.2.4 CCC algorithm

The CCC algorithm can be outlined as follow: at each time step, the algorithm is initialized, then a loop is executed: for all the combinations of line J_c in J_{C0} (subset of the constraints of J_C that may be violated at the next time step), a solution is computed and *scaled* by the most constraining constraint. The loop stops when all the combinations have been tried or when the error is lower than the threshold. The algorithm is illustrated by Fig. 3.11. The following of this section details the implementation.

3.2.4.1 Jacobian inversions and implementation

This part describes general implementation elements. A dedicated section (4.1.1.2) focuses on the practical implementation of the algorithms (numerical values etc.).

[1] Considerations about possible perspectives are presented in sections 3.4 and 5.2.2.

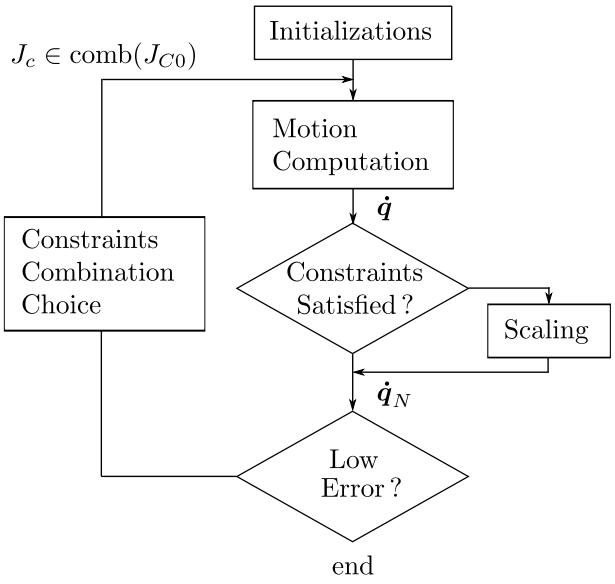

Figure 3.11: CCC Algorithm. $\text{comb}(J_{C0})$ denotes the set of combinations of lines of J_{C0}

3.2.4.1.1 Pseudo-inversions In the control law of Eq. (3.21), in order to avoid the usual conditioning problems in the neighborhood of singularities, the pseudo-inversions are done with the Damped Least Square (DLS) method (matrix conditioning improvement, see Nakamura *et al.* (1986) and appendix B.2.2).

$$\begin{aligned}\dot{\boldsymbol{q}} =& (J_O P_{J_c})^{+,DLS} \dot{\boldsymbol{X}}_{des,O} \\ &+ \left(J_{\bar{c}}\, P_{\begin{bmatrix} J_c \\ J_O \end{bmatrix}} \right)^{+,DLS} (\dot{\boldsymbol{X}}_{des_{\bar{c}}} - J_{\bar{c}} (J_O P_{J_c})^{+,DLS} \dot{\boldsymbol{X}}_{des,O})\end{aligned} \tag{3.22}$$

3.2.4.1.2 Projectors A common way of computing a projector P_A is given by $P_A = (I - A^+ A)$, which can be made robust to singularity thanks to the DLS inversion $P_A = (I - A^{+,DLS} A)$; however, the DLS method induces an error in the inversion that distorts the projection. As a consequence, the tasks of lower priority may have an impact on the fulfilment of tasks of upper priority, which is not acceptable in our case, as constraints may be violated. A safe way to compute P_A can be obtained directly from the Singular Value Decomposition (SVD) of A which provides an access to the kernel of A without requiring its inversion (see appendix B.3.2 and Baerlocher *et al.* 2004)

$$P_A = V_{r,n} V_{r,n}^T \tag{3.23}$$

where V is issued from the SVD decomposition of A and $V_{r,n}$ is the matrix which content is the r to n columns of V, r being the rank of A.

For the same reason, in the terms $(AP_B)^{+,DLS}$, the projection is biased by the DLS, which once again may provokes an impact of lower priority tasks to upper priority tasks. To prevent this, a preprojection is introduced $(AP_B)^{+,DLS} \rightarrow P_B (AP_B)^{+,DLS}$ (see appendix B.3.2).

$$\begin{aligned}\dot{\boldsymbol{q}} =& P_{J_c} (J_O P_{J_c})^{+,DLS} \dot{\boldsymbol{X}}_{des,O} \\ &+ P_{\begin{bmatrix} J_c \\ J_O \end{bmatrix}} \left(J_{\bar{c}}\, P_{\begin{bmatrix} J_c \\ J_O \end{bmatrix}} \right)^{+,DLS} (\dot{\boldsymbol{X}}_{des_{\bar{c}}} - J_{\bar{c}} P_{J_c} (J_O P_{J_c})^{+,DLS} \dot{\boldsymbol{X}}_{des,O})\end{aligned} \tag{3.24}$$

3.2.4.1.3 Distance computation The distance computation is provided by a collision detection package[2]. As for most of these tools, the information returned is, for each segment, the point of the segment that is the closest to the obstacles. Anyway, the approach that consists in constraining only the closest point to the obstacle (based on Maciejewski *et al.* (1985) for the whole manipulator and on Faverjon *et al.* (1987) for convex segments) seems valid in continuous time. In discrete time, progressive displacements can violate constraints because of segment rotations around the constrained point. To our knowledge, no work has been carried out on the consequences of the transition from continuous to discrete time control on obstacle avoidance, although a contribution of Kanehiro *et al.* (2008) deals with the passage of strictly convex to non-strictly convex segments, in which the problems are similar to those encountered when passing from continuous to discrete time. To overcome these drawbacks, Peinado *et al.* (2009) propose an efficient iterative method using preventive damping constraints in case of potential collisions.

3.2.4.2 Particular case of the joint velocity limits - Scaling

It is not appropriate to address joint velocity limits with passive avoidance as it may stop a joint to prevent it from going too fast. If a joint velocity is too high, it is preferred to *scale* the control solution to the maximal admissible velocity, *i.e.* to reduce the norm of the control solution to fit the velocity limit. To maintain the motion operational coherency, the scaling is done *homogenously*, *i.e.* with a proportionality coefficient applied over the whole vector, so that the operational direction of the solution is maintained.

$$\dot{\boldsymbol{q}}_N = \dot{\boldsymbol{q}} \min_{0 \leq i \leq n} \left(\frac{\dot{q}_{i,max}}{||\dot{q}_i||} \right) \tag{3.25}$$

where $\dot{q}_{i,max}$ is the maximum velocity of joint i.

3.2.4.3 Admissibility and scaling generalization

The test of admissibility with respect to the constraints is performed through

$$J_C \dot{\boldsymbol{q}} \leq \boldsymbol{b} \tag{3.26}$$

where $\boldsymbol{b}$ is the concatenation of the maximum velocities along the constraints. In order to increase the compliance of the solutions with respect to the constraints, the scaling technique is extended to all the constraints and applied to all the control solutions obtained.

$$\dot{\boldsymbol{q}}_N = \dot{\boldsymbol{q}} \min_{\substack{0 \leq i \leq p \\ (J_C \dot{\boldsymbol{q}})_i > 0}} \left(\frac{\boldsymbol{b}_i}{(J_C \dot{\boldsymbol{q}})_i} \right) \tag{3.27}$$

where $\min_{\substack{0 \leq i \leq p \\ (J_C \dot{q})_i > 0}}$ denotes the minimum of the given expression for all the constraints (i between 1 and p) along which the joint motion is positive ($(J_C \dot{q})_i > 0$). The scaled motion $\dot{\boldsymbol{q}}_N$ generalizes the one of Eq. (3.25). In order to limit the notation, at the end of the admissibility test, $\boldsymbol{q}$ gets the value of $\boldsymbol{q}_N$

$$\dot{\boldsymbol{q}} \leftarrow \dot{\boldsymbol{q}}_N \tag{3.28}$$

[2]SWIFT++: Speedy Walking via Improved Feature Testing for Non-Convex Objects. Ehmann *et al.* (2001). http://gamma.cs.unc.edu/SWIFT++

The systematic scaling enables to obtain an admissible solution for each constraints combination in J_c. Actually, as the null motion belongs to the intersection of the control admissibility spaces ($\mathscr{C}(\boldsymbol{\sigma})$, see section 2.3.1), then for any solution

$$\forall\ \dot{\boldsymbol{q}} \in \mathbb{R}^n,\ \exists\ \alpha \in [0,1],\ \dot{\boldsymbol{q}}_{(\alpha)} = \alpha\dot{\boldsymbol{q}} \text{ is admissible} \tag{3.29}$$

In fact, as the constraints are all linear inequalities (as in Eq. (3.26)), the solutions space is convex and contains the null solution (no motion). So, in every direction of the joint displacement space, there exists an admissible solution, which norm is null in the worst case. This method enables to get as close as possible to the constraints if there is a persistent demand in that way (input repetition).

3.2.4.4 Algorithm

The general CCC algorithm is given by Algorithm 3.

Algorithm 3 : CCC

$J_C \leftarrow$ constraints under the active avoidance threshold
$J_{C0} \leftarrow$ constraints of J_C that may be violated at the next time step (tight threshold)
$\dot{\boldsymbol{q}}_{end} \leftarrow \mathbf{0}$ rad; $\dot{\boldsymbol{q}} \leftarrow \mathbf{0}$ rad
$\text{Err}_{end} \leftarrow 1$ m/s; $\text{Err} \leftarrow 0$ m/s; $\epsilon \leftarrow 10^{-2}$ m/s;
for all Lines Combinations J_c in J_{C0} **do**
  $\dot{\boldsymbol{q}} \leftarrow$ Joint velocity Computation - Eq. (3.24)
  Admissibility Test - Eq. (3.26)
  Scaling - Eqs. (3.27), (3.28)
  $\text{Err} \leftarrow ||J_O\dot{\boldsymbol{q}} - \dot{\boldsymbol{X}}_{des,O}||$
  if $\text{Err} \leq \epsilon$ **then**
    break
  else
    if $\text{Err} \leq \text{Err}_{end}$ **then**
      $\text{Err}_{end} \leftarrow \text{Err}$; $\dot{\boldsymbol{q}}_{end} \leftarrow \dot{\boldsymbol{q}}$
    end if
  end if
end for
send $\dot{\boldsymbol{q}}_{end}$

Once the initialization is done (the variables with subscript $_{end}$ being the data to be sent at the end of the time step), the loop over the combinations of lines of J_{C0} put in J_c begins. Based on the admissibility test which evaluates the highest constraint violation rate, the scaling reduces the magnitude of the solution so that it complies with the constraints. If the operational error obtained from the resulting joint velocity is acceptable (the value of $\epsilon \leq 10^{-2}$ m/s is an example), then the computation is finished and $\dot{\boldsymbol{q}}_{end}$ is sent; if not, the combination of constraints of J_{C0} is changed in J_c and a new iteration begins. If all the combinations have been tried, the joint velocity yielding the minimum operational error is sent to the actuators.

For complexity reasons, the condition on the combinations *For all lines combinations J_c in J_{C0}* can be approximated to the addition of the jacobians of violated constraints at each iteration to J_c (being initialized empty). When all lines of J_{C0} are in J_c, then the most constrained motion is carried out, which necessarily produces a constraint

compliant motion, and can thus constitute a condition to get out of the loop. In this case, the number of iteration is limited to the number of lines of J_{C0}. This is the implementation carried out in chapter 4.

3.2.5 Partial conclusion and CCC theoretic performances

The *CCC* is an iterative control method that solves the hierarchical multi-objective control problem while satisfying any number of static constraints of obstacles, joint boundaries and joints velocity limits.

In non over-constrained cases (when the robot is able to fulfil all the objectives and all the active avoidance terms), the CCC behavior is equivalent to the control law of Maciejewski *et al.* (1985) which is optimal and very fast. In a more global scope, the behavior of a robot submitted to this control law tends to maintain these non over-constrained situations as long as they are possible (it tends to get far from the constraints). In over-constrained cases, the CCC has the advantage to always satisfy the constraints (assumed to be compatible). However, in these cases, the CCC may be non optimal. Fig. 3.12 illustrates the comparison between an active set method and the CCC method. The solution found by the CCC is not optimal, and a strong perspective is to rely on the KKT optimality conditions to increase efficiency and optimality. However, as mentioned previously, this comparison is partial, as the main advantage of the CCC is its satisfying behavior in cluttered environments.

3.3 Displaced configuration control

In this section, a technique that fully takes advantage of the constraints formulation is proposed. It resorts to a virtual configuration called *displaced configuration* to adapt the usual operational control scheme to satisfy any kind of compatible constraints.

For example, when the null motion is not admissible with respect to the constraints (in other words, the current e-state is not an instantaneous safe e-state for the next time step), techniques to compute a constraint compliant motion such as scaling (section 3.2.4.3) cannot be used: the current configuration being not constraint compliant, there may not be any α in Eq. (3.29) such that a constraint compliant solution can be found (see Fig. 3.13).

One can observe that the scaling can be applied safely to any control solution computed from an instantaneous safe e-state (the null control vector being constraint compliant in that case, $\alpha = 0$ in Eq. (3.29) produces a constraint compliant control vector). The idea of the displaced configuration is to solve the control problem from another configuration $\boldsymbol{q}_{dp}$ than the current one $\boldsymbol{q}$, such that the scaling can be used. To that end, $\boldsymbol{q}_{dp}$ should be chosen so that the resulting displaced e-state (σ with $\boldsymbol{q}_{dp}$ instead of $\boldsymbol{q}$) is instantaneous safe.

This control law principle can be used in various frameworks:

- It extends the CCC framework to deal with any kind of constraints;
- It enables to obtain safe single iteration resolution scheme in presence of compatible constraints.
- It is a solution of the *phase I problem* (finding an initial point admissible with respect to the constraints) in convex optimization;

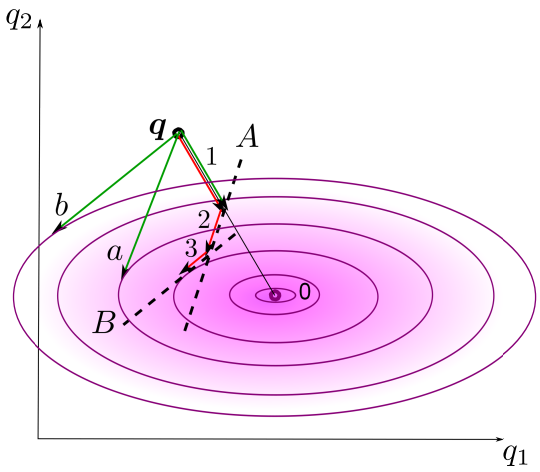

Figure 3.12: Comparison of an active set method and the CCC method on a 2 DOFs problem constrained by 2 inequalities (A and B). The active set methods, assumed to begin with an empty active constraint set, needs 3 iterations to reach the optimal point (motions 1, 2 and 3). The CCC method is not an *incremental* method, *i.e.* during one time step it does not concatenate several control vectors obtained through several iterations (as vectors 1, 2, and 3 for the active set method) to obtain the final control solution; each iteration returns a whole control solution. It relies on scaling to move progressively, on several time steps, toward the goal. The CCC, assumed to begin with an empty passive avoidance set, has the same result of the active set method on the first iteration (motion 1). Then, the constraint A is avoided passively (motion a), and then constraint B (motion b). Avoiding constraint A and B simultaneously produces no motion (2 DOFs problem). The retained motion is a, as it reaches the closest equipotential curve.

This section is divided as follows: first, the displaced configuration based control law is exposed. Then, based on a usual IVK reactive control scheme, a full control algorithm using displaced configuration is detailed, relying on scaling to comply with any kind of compatible constraints. Finally, the application of the displaced configuration within a single iteration constraint compliant control law is exposed.

3.3.1 Displaced configuration based control law

For clarification, the space $E_C(\boldsymbol{\sigma})$ is introduced. $E_C(\boldsymbol{\sigma})$ denotes the subspace of the joint space containing all the configurations resulting from all the admissible control vectors ($\mathscr{C}_A(\sigma)$). As the constraints are assumed to be linear, and given the approximation of finite differences between time steps, $E_C(\boldsymbol{\sigma})$ is convex (intersection of semi intervals). $E_C(\boldsymbol{\sigma})$ is never empty as the constraints are assumed to be compatible (else an ASB is used and the control problem resolution is out of the scope of this chapter).

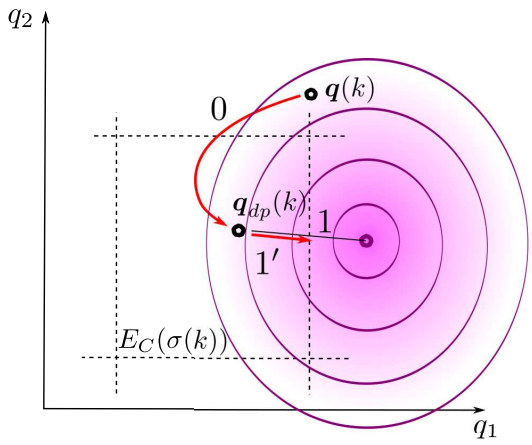

Figure 3.13: Use of a displaced configuration. The current configuration $\boldsymbol{q}(k)$ is virtually displaced (0) to the admissible configuration $\boldsymbol{q}_{dp}(k)$, from which a solution is computed (1). This solution can then be scaled safely (1'). The final motion is the vector represented by (0)+(1').

3.3.1.1 Displaced configuration expression

The basis of the displaced configuration is to work from another configuration $\boldsymbol{q}_{dp}(k)$ than the current one $\boldsymbol{q}(k)$. The motion is then divided into two terms

$$\dot{\boldsymbol{q}} = \underbrace{\frac{\boldsymbol{q}_{dp} - \boldsymbol{q}}{\delta t}}_{\text{Reaching the displaced configuration}} + \underbrace{\dot{\boldsymbol{q}}_{dp}}_{\text{Motion toward the target}} . \tag{3.30}$$

- the first term is the displacement toward the displaced configuration. This solution is always constraints compliant, as the displaced configuration belongs to $E_C(\boldsymbol{\sigma})$;
- the second term is the motion from the displaced configuration toward the target.

When using the scaling technique (section 3.2.4.3) to comply with constraints for example, the scaling is only applied on the second term

$$\dot{\boldsymbol{q}} = \frac{\boldsymbol{q}_{dp} - \boldsymbol{q}}{\delta t} + \alpha \dot{\boldsymbol{q}}_{dp}. \tag{3.31}$$

where α is the scaling rate.

3.3.1.2 Choice of the displaced point

The choice of $\boldsymbol{q}_{dp}(k)$ in $E_C(\sigma(k))$ at the beginning of a time step determines the behavior of the robot and may impact the optimality of the motion (see Fig. 3.14). Potentially, all configurations of $E_C(\boldsymbol{\sigma})$ can be retained as the displaced configuration.

The elements to take into account for this choice are the considered constraints and predictability of the operational inputs:

- When the constraints are compatible and without any information on the operational inputs at the next time step, a good compromise is to take $\boldsymbol{q}_{dp}(k)$ at the

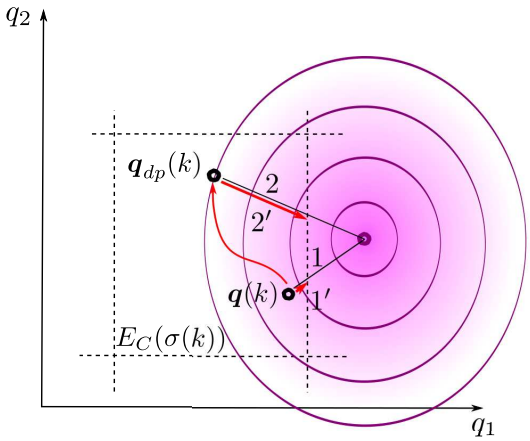

Figure 3.14: One of the interests of the displaced configuration method when using scaling. A configuration ($\boldsymbol{q}(k)$) close or "on" a constraint induce a bad scaling rate ($\alpha = ||1||/||1'||$ tends to 0). If the above representation is a small part (projection) of the whole problem (robot with more than the two represented DOFs), the whole motion (and thus the operational error) may suffer from this bad scaling rate. When using a displaced configuration ($\boldsymbol{q}_{dp}(k)$), the solution (2) permits to obtain a correct scaling factor ($\alpha = ||2||/||2'||$) that, even if non optimal, enables a significant motion.

center of $E_C(\boldsymbol{\sigma})$. If the considered constraints are exclusively joint-dependent (*e.g.* joint position, velocity, acceleration, torque limits), for each articulation i it yields

$$q_{dp,i} = \frac{q_{(dp,m),i} + q_{(dp,M),i}}{2} \tag{3.32}$$

where $\boldsymbol{q}_{dp,M}$ (respectively $\boldsymbol{q}_{dp,m}$) is the vector of maximum (respectively minimum) joint position of $E_C(\boldsymbol{\sigma})$. This choice leaves an equal space of admissible motion on both side of $q_{dp,i}(k)$. It is particularly appropriate when using scaling to keep some admissible space around $\boldsymbol{q}_{dp}(k)$ as the motion is scaled by the maximum violation rate of the motion beginning at $\boldsymbol{q}_{dp}(k)$ with respect to the constraints limits.

- When the constraints are not proved to be compatible (as in section 2.4: obstacles, joint position, velocity and acceleration limits), a relevant possibility is to place $\boldsymbol{q}_{dp}$ at the maximal deceleration configuration as it necessarily belongs to $E_C(\boldsymbol{\sigma})$ (cf. section 2.4.5.1)

$$\boldsymbol{q}_{dp} = \boldsymbol{q}_{\text{decM}}. \tag{3.33}$$

On the one hand, it favors joint accelerations in the direction of the previous velocity which constitutes an *a priori* on the operational tasks. On the other hand, it is less efficient for operational motions with sudden direction changes. As a remark, this displaced configuration is a solution of the phase I problem in convex optimization. The phase I problem aims at finding an initial constraints compliant solution from which to work (the phase II problem being the optimization of this admissible solution). Even if the problem is known to be feasible, the phase I problem can be hard to solve and constitutes an optimization algorithm in itself. In the framework of chapter 2, the permanent availability to the developed ASBs is always ensured, which guarantees that a maximum deceleration is always admis-

sible (section 2.4.5). As a result, taking the maximal deceleration configuration $\boldsymbol{q}_{\mathrm{decM}}(k)$ as the displaced configuration offers a solution to the phase I problem.

3.3.2 Modification of a usual IVK reactive control scheme

This section details the modifications induced by the use of a displaced configuration when the technique to comply with constraints is scaling (section 3.2.4.3).

3.3.2.1 Usual IVK reactive control scheme

The scheme of a usual reactive control law algorithm is described in algorithm 4. First, the operational input is computed based on the measures of the current e-state and the desired behavior. Second, an IVK algorithm is used to compute joint velocities from the operational input. Third, the joint velocity is treated to comply with constraints not handled in 2 (by scaling for example). Fourth, the resulting joint velocity is sent to the lower control loop.

Algorithm 4 : General operational space control law scheme

1. Computation of $\dot{\boldsymbol{X}}_{des,O}$ from $\boldsymbol{X}_{des,O} - \boldsymbol{X}_O$
2. Jacobian Inversion: $\dot{\boldsymbol{q}} = IVK(J, \dot{\boldsymbol{X}}_{des,O})$
3. Operations to comply with the constraints not handled in 2.
4. Send $\dot{\boldsymbol{q}}$

3.3.2.2 Modified control algorithm

Using a displaced configuration modifies Algorithm 4 into Algorithm 5.

Algorithm 5 : Control law iteration using the displaced configuration

1. Choice of $\boldsymbol{q_{dp}} \in E_C(\boldsymbol{\sigma})$; $\boldsymbol{X}_{O,dp} = G_O(\boldsymbol{q}_{dp})$
2. Computation of $\dot{\boldsymbol{X}}_{des,O,dp}$ from $\boldsymbol{X}_{des,O} - \boldsymbol{X}_{O,dp}$
3. Jacobian Inversion: $\dot{\boldsymbol{q}}_{dp} = IVK(J, \dot{\boldsymbol{X}}_{des,O,dp})$
4. Operations to comply with the constraints: $\dot{\boldsymbol{q}}_{dp,(\alpha)} =$ scaling$(\dot{\boldsymbol{q}}_{dp})$
5. Send $\dot{\boldsymbol{q}} = \frac{\boldsymbol{q}_{dp} - \boldsymbol{q}}{\delta t} + \dot{\boldsymbol{q}}_{dp,(\alpha)}$ to low level controller

3.3.2.3 Step 1: Computation of the displaced configuration and the displaced operational position

Considerations about the choice of $\boldsymbol{q}_{dp}$ are exposed in section 3.3.1.2. Basically, as the solution is scaled, and as the scaling is coherent, it is important to keep a scaling factor (α in Eq. (3.29)) as high as possible to keep a significant part of the motion (see Fig. 3.14). Thus, the displaced configuration should be chosen as far as possible from the most violated constraints.

The computation of $\boldsymbol{X}_{O,dp}$ is obtained by applying the forward kinematic model $G_O()$ to $\boldsymbol{q}_{dp}$.

3.3.2.4 Step 2: Computation of the desired displaced operational velocity

$\dot{\boldsymbol{X}}_{des,O,dp}$ is computed similarly to $\dot{\boldsymbol{X}}_{des,O}$ in the initial algorithm, but taking $\boldsymbol{X}_{O,dp}$ instead of $\boldsymbol{X}_O$.

3.3.2.5 Step 3: Inverse Velocity Kinematics

This step aims at computing the joint velocities $\dot{\boldsymbol{q}}_{dp}$ that minimizes the error $||\dot{\boldsymbol{X}}_{des,O,dp} - J\dot{\boldsymbol{q}}_{dp}||$. The Jacobian is assumed[3] to be constant between $\boldsymbol{q}$ and $\boldsymbol{q}_{dp}$, so no Jacobian computation is done at configuration $\boldsymbol{q}_{dp}$.

As mentioned in the introduction, any kind of IVK method (CCC, COM, AIM) can be used with various advantages. For any technique relying on Jacobians inversions submitted to exclusively joint dependent constraints (*e.g.* joint position, velocity, acceleration and torque limits), the following indication is proposed: as the scaling rate is aimed to be maximized, in order to take into account the motion capabilities of each joint at the current time step and thus reduce the resort to highly constrained joints (see Fig. 3.15), the inverse velocity kinematics is weighted in the joint space (see appendix B.3.1) by the width of each joint admissibility interval

$$M_q^{-1} = \text{diag}(\boldsymbol{q}_{dp,M} - \boldsymbol{q}_{dp,m}) \tag{3.34}$$

where M_q^{-1} is the weighting matrix used for the inversion (see Eq. (B.8) of appendix B.3.1). This technique enables to balance the joints contributions to the operational motion with respect to their current capabilities. Moreover, it has the interesting effect to prevent a joint motion when the associated weighting coefficient is null (semi-positive definite matrix weighting, see section 3.1.3.3). For example, when the constraints impose a maximal joint deceleration for a given articulation i, the associated admissibility interval is reduced to a point (which is necessarily the displaced point): so, the interval width is null and the inversion does not take this joint into account. The only motion is then to reach the displaced configuration (cf. Eq. (3.31)). As a result, the joint is not asked to contribute to the motion ($\dot{q}_{i,dp} = 0$ in Eq. (3.31)) and the scaling is not impacted by the proximity of this joint to its constraint (α can take any value in Eq. (3.31)).

3.3.2.6 Step 4: Compliance with the constraints

This step aims at making the solution calculated at step three $\dot{\boldsymbol{q}}_{dp}$ compliant with the constraints. This operation is done by scaling, and as mentioned previously, it is important to make it from of $\boldsymbol{q}_{dp}$ as it belongs to $E_C(\boldsymbol{\sigma})$. The general constraints expression is

$$J_C\dot{\boldsymbol{q}} \leq \boldsymbol{b} \tag{3.35}$$

where J_C is the concatenation of all the Jacobian constraints and $\boldsymbol{b}$ the concatenation of all associated limits. The introduction of $\dot{\boldsymbol{q}}_{dp}$ yields (cf. Eq. (3.31))

$$J_C\dot{\boldsymbol{q}}_{dp} \leq \boldsymbol{b} - J_C\frac{\boldsymbol{q}_{dp} - \boldsymbol{q}}{\delta t} \triangleq \boldsymbol{b}_{dp} \tag{3.36}$$

[3] As an example, a 1-DOF rotational joint submitted to acceleration limits and moving at $\dot{q} = 1\ rad/s$ with a $\delta t = 10\ ms$ time step has a displacement $q_{dp} - q = 0.01\ rad$ when taking the displaced configuration of Eq. (3.32). The resulting impact on the Jacobian is then very low, even in the neighborhood of singularities where the use of Damped Least Square inversion induces errors much higher.

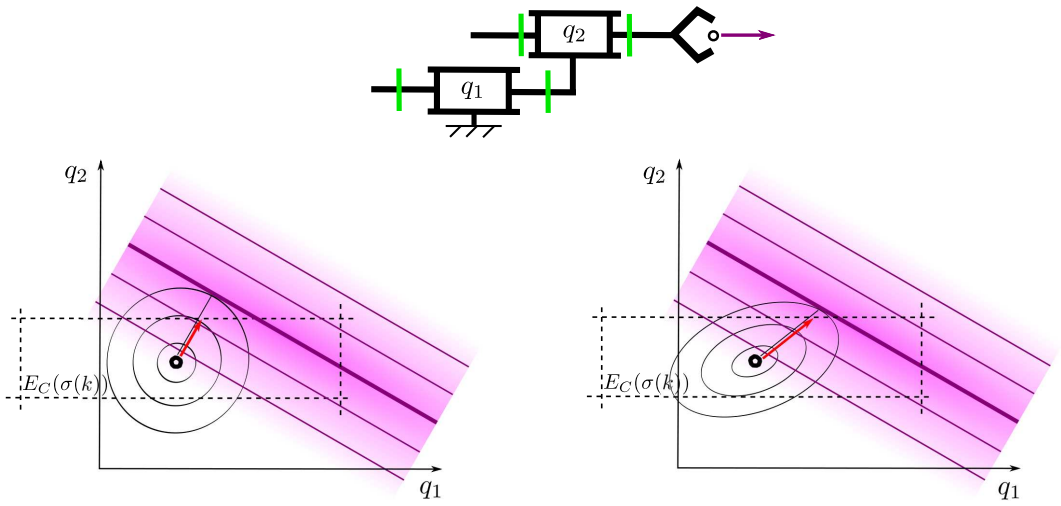

Figure 3.15: Interest of the pseudoinverse weighted by the joint admissibility interval. In this redundant case, the system is assumed to be only limited by its joint position limits (in green on the upper scheme). On the left graph, the representation of the problem in the joint space shows that a direct inversion without particular weighting minimizes the Euclidian norm (equipotential curves are circles) which, given the constraints limits positions, yields a bad scaling. Conversely, a relevant weighting (right graph) enables to obtain a better scaling, and thus a better solution. In this example, a more accentuated weighting would have certainly led to an optimal solution (no scaling needed).

Then, if Eq. (3.36) is not verified, the scaling is done by (cf. Eq. (3.27))

$$\dot{\boldsymbol{q}}_{dp,(\alpha)} = \dot{\boldsymbol{q}}_{dp} \min_{\substack{0\leq i\leq j \\ (J_C\dot{q})_i>0}} \left(\frac{\boldsymbol{b}_{dp,i}}{(J_C\dot{\boldsymbol{q}})_i} \right) \tag{3.37}$$

where j is the number of constraints.

3.3.3 Single iteration constraints compliant control law SICCC

The displaced configuration control scheme can be used to solve efficiently the control problem in a single iteration, thus offering a fast resolution method. Actually, the need to resort to iterations in constraints compliant control laws comes from the fact that it is not possible to know *a priori* when a motion is forbidden or not. Let a manipulator being "on" one of its joint position limits. If the associated constraint is not active at the beginning of the resolution (as it is necessarily the case in clamping or CCC), and if the motion pushes the joint over its boundary, the violation rate is infinite and the scaling step cuts this first solution to 0, which prevents any progression of the joints, even the unconstrained ones. The algorithm then begins again the resolution, with the considered constraint avoided passively. Conversely, with a displaced configuration taken far from the constraints, motions in every directions are possible, thus the exceedance rate cannot become infinite, the first control solution computed is not null (see Fig. 3.14) and the scaling step enables a reasonable progression for the joint not concerned by the constraints in a coherent motion.

As mentioned in section 3.3.2.5, any kind of inversion method can be used. A control law example can then be adapted from the 2-levels multi-objective control law

of Maciejewski *et al.* (1985)

$$\dot{\boldsymbol{q}}_{\boldsymbol{dp}} = J_O^{\#,DLS}\dot{\boldsymbol{X}}_{des,O,dp} + P_{J_O}(J_C P_{J_O})^{\#,DLS}(\dot{\boldsymbol{X}}_{des,C,dp} - J_C J_O^{\#,DLS}\dot{\boldsymbol{X}}_{des,O,dp}) \quad (3.38)$$

where $^{\#,DLS}$ is the operator of weighted pseudo-inversion with Damped Least Square approximation. Even if the initial control law of Maciejewski is not safe, the compliance to constraints is ensured here by the scaling.

This control law offers a good compromise between quality and efficiency. It can be seen as the first step of an active-set method algorithm with a chosen phase I solution (cf. appendix C). It has the advantage to offer good solutions to the inequality constrained control problem in a single iteration scheme.

However, this control laws sometimes implies a deterioration of the task performances. For example, this is the case of Fig. 3.16. This case happens when the optimal configuration does not belong to the displaced configuration/ideal configuration progression line. This drawback can be cancelled by a simple test to stay on the current (and thus better) configuration. Even if not treated by a test, this motion converges toward a stable configuration (no oscillation) if the retained displaced configuration is one of the two proposed in section 3.3.1.2.

3.4 Partial conclusion and perspectives

This chapter deals with the control problem resolution of a feasible problem (for example issued from chapter 2).

Based on state-of-the-art methods, the Constraints Compliant Control is set up to obtain a satisfying behavior in presence of strict priority levels and imperative constraints. The notion of passive avoidance is introduced to enforce the satisfaction of an arbitrary number of constraints; a common strict priority multi-objective structure is used for the tasks to carry out and the low priority level is reserved for active avoidance tasks in order to maintain a good problem conditioning and avoid behavior singularities due to multiple constraints closeness. Iterations on the constraints avoided passively and solutions scaling are used to increase optimality.

The CCC scope is limited to problems involving constraints allowing an instantaneous stop. It is extended to any kind of compatible constraints through the use of a displaced configuration. This technique can also be used in a safe single iteration control law offering a compromise between efficiency and optimality.

The control scheme obtained as an outcome of chapter 2 and 3 is presented on Fig. 3.17

Perspectives involve the improvement on the CCC thanks to the exploitation of the KKT conditions to enable an intelligent exploration of the constraints combination space. This exploration could also be guided by factors related to the global mission (*e.g.* active avoidance for better motion capabilities, passive avoidance to work close to the constraints) and not only the performance of the task at the current time step. On another scope, the active avoidance of obstacles could be influenced by the convexity parameters of the environment and the time step period or the velocity of the robot (*e.g.* the work on collidability of Choi *et al.* (2000)).

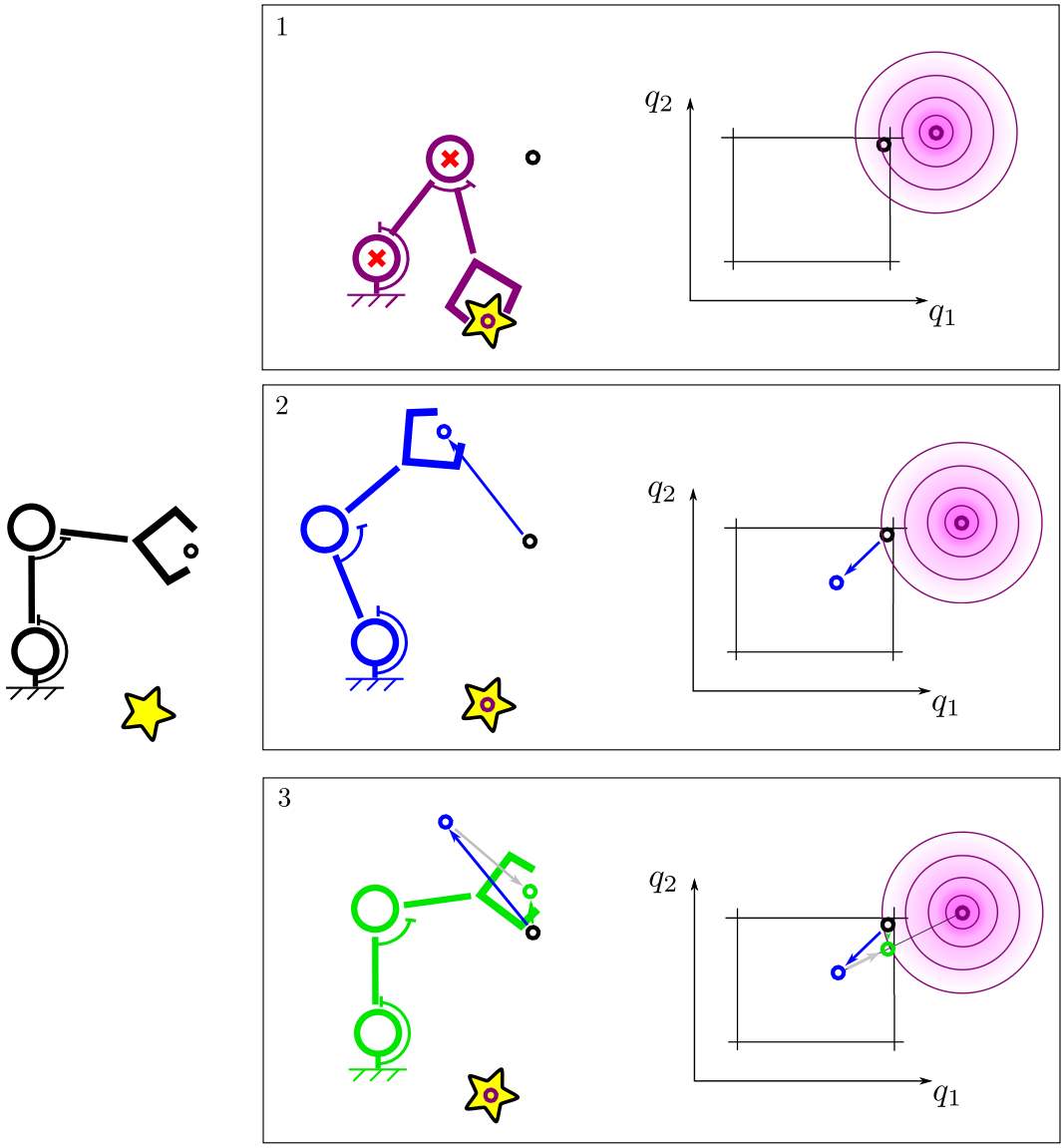

Figure 3.16: Illustration of a deterioration phenomenon. Left: the robot in the current e-state, trying to reach a star. (1): solution (not admissible) in the unconstrained case. (2): displaced configuration: the configuration gets far from the constraints; (3): final solution, the final motion gets away from the objective (its configuration is on a further equipotential). Nonetheless, the resulting behavior tends to converge to a stable configuration.

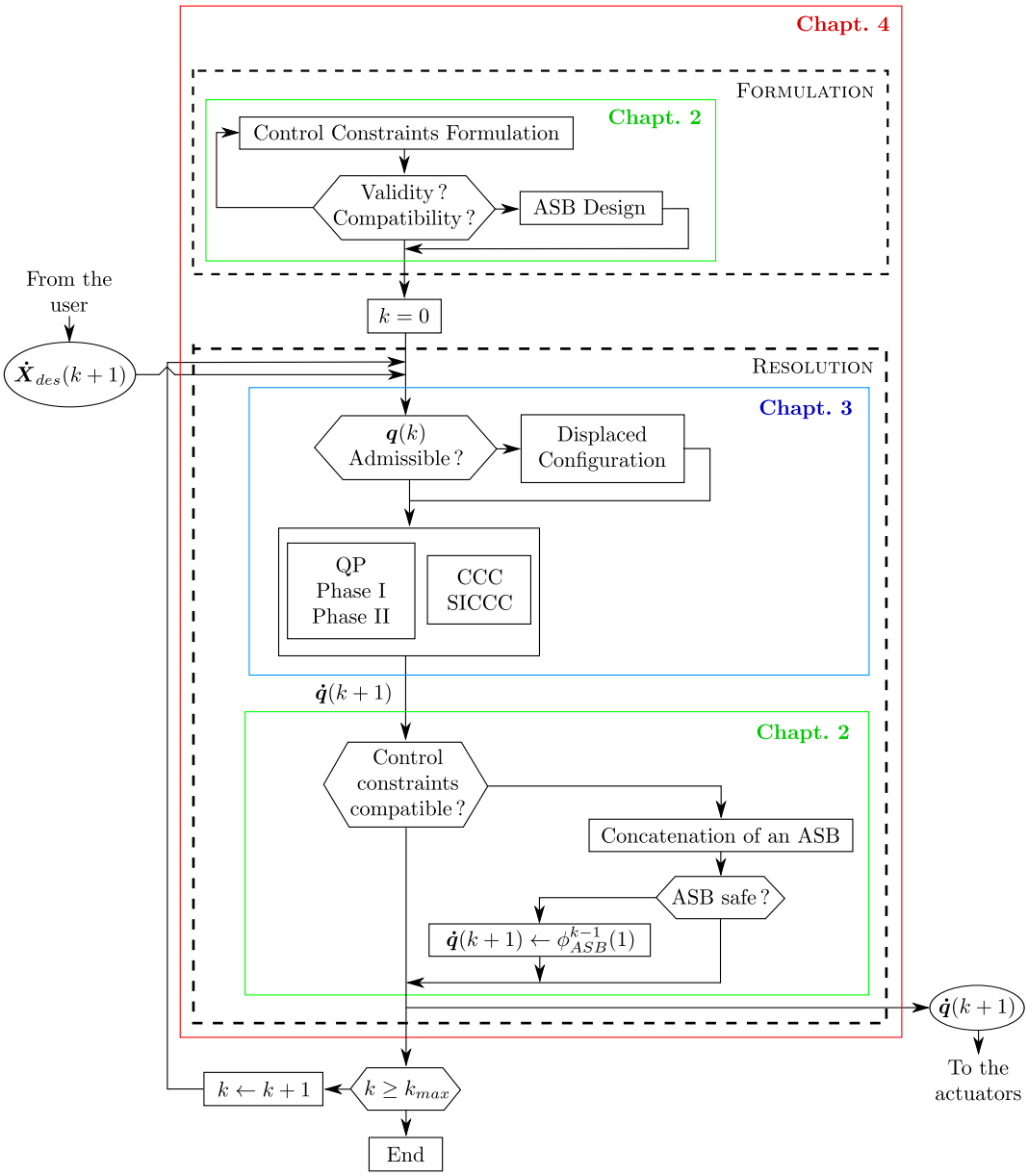

Figure 3.17: **Safe Constraint Compliant Controller.** From each identified e-state constraint (physical limit or induced by the mission), a control constraint is formulated offline. If the validity and the compatibility of the control constraints cannot be proved, a new formulation can be expressed and evaluated, or an ASB must be established. Once this is done, the reactive control loop is launched. At each time step, the controller is fed with operational inputs. If the current configuration is not admissible, working from a displaced configuration enables to maintain constraints compliance with any control method and solves the phase I of a Quadratic Program. If the control constraints defined offline could not be proved valid and compatible, an ASB sequence is concatenated to the desired joint motion: if the resulting behavior is not safe, then the first control input of the ASB is sent, which safety has been proved at a previous time step; else, the control solution is sent to the actuators.

Chapter 4

Results

The work carried out along this thesis is involved in the TELEMACH project which addresses the feasibility of replacing human interventions in TBMs excavation rooms by teleoperated maintenance (see chapter 1). The retained missions (inspection, cleaning, tools replacement) would be performed by dedicated robotic systems: articulated arms for inspection and cleaning, tools changer, heavy loads manipulator (for instance, each disc cutter tool weighs 150kg), conveyor and automated airlock doors.

The environment is complex and the interventions are critical, so both the design and the control of the manipulator dedicated to inspection and cleaning have been addressed during the thesis:

- The manipulator design is carried out thanks to global multi-objective optimization techniques (evolutionary algorithm); in this context, a huge number of robots are evaluated through a trajectory tracking simulation, for which a specific control law is required;

- The control of a teleoperated manipulator in real time in a TBM cutter head demands an extreme level of safety, as any failure is critical (impossibility for human operators to intervene), which is similar to the nuclear plant dismantling context. A special attention to the constraints compatibility is meaningful for these applications.

This chapter is divided in 3 sections. In the first section, simulations obtained with the CCC and the displaced configuration method are presented; these simulations validate the CCC approach and its application to the evolutionary design process exposed in the second section. Finally, the third section describes the experiments carried out with a 6-DOF manipulator in real-time trajectory tracking for compatible and incompatible constraints sets.

4.1 CCC and displaced point control simulation results

This section describes the results obtained in simulation with the control laws developed in chapter 3. First, the CCC is compared to other state-of-the-art control laws in the constraint context exposed in the first case study (section 2.4.2). It can be considered as a preliminary validation for the application of the CCC to the evolutionary design process presented in section 4.2. Then, the second section exposes results obtained with

the displaced configuration control used both in a single iteration control law and in the CCC structure, in the constraints context of the second case study (section 2.4.3).

4.1.1 CCC results

This section is dedicated to the simulation of the CCC in presence of the following constraints:

- Joint position limits - control constraints Eqs. (2.21) and (2.22);
- Joint velocity limits - control constraints Eqs. (2.23) and (2.24);
- Obstacles - control constraints (2.25).

As established in section 2.4.2, this set is compatible.

4.1.1.1 Simulations presentation

The proposed simulations consist of two inspection missions involving trajectories close to obstacles. The proposed environment is composed of a column and a wall; the manipulator has 6 DOFs, all the links being 1 DOF rotational joints. The environment and the manipulator are represented on Fig. 4.1.

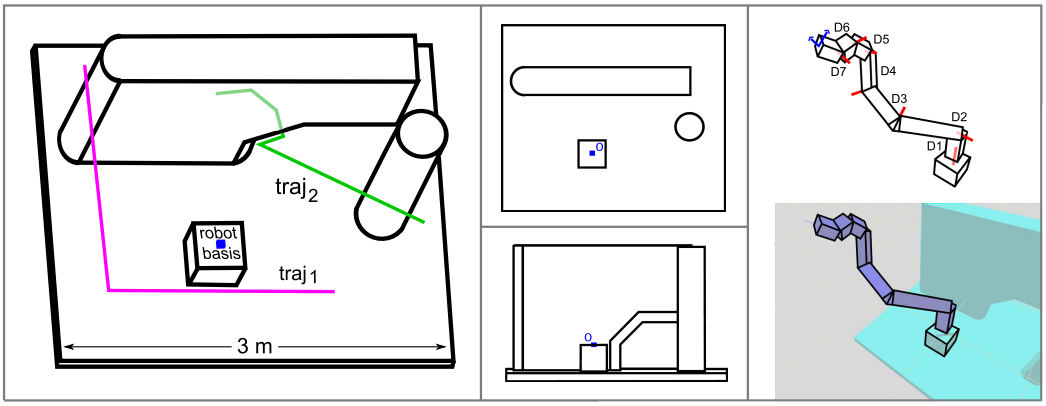

Figure 4.1: Views of the environment and the two trajectories to track; manipulator schemes.

To assess the performances of the CCC, 3 multi-objective control laws are compared:

- Control law A: Trajectory tracking as 1^{st} task, obstacle avoidance as 2^{nd} task (adapted from Maciejewski *et al.* (1985)):

$$\dot{\boldsymbol{q}} = J_O^{+,DLS}\dot{\boldsymbol{X}}_O + P_{J_O}(J_C P_{J_O})^{+,DLS}(\dot{\boldsymbol{X}}_C - J_C J_O^{+,DLS}\dot{\boldsymbol{X}}_O) \tag{4.1}$$

with J_O the Jacobian of the effector for trajectory tracking, $J_O^{+,DLS}$ its DLS-pseudoinverse (see appendix B.2.2), P_{J_O} an exact projector on the kernel of J_O based on the SVD decomposition of J_O (see appendix B.3.2), and J_C the Jacobian of obstacle avoidance. $\dot{\boldsymbol{X}}_O$ is the trajectory tracking desired velocity and $\dot{\boldsymbol{X}}_C$ the avoidance terms based on the inverse of the distance to the closest points of each segment to the obstacles. The "adaptation" from the control law of Maciejewski

consists of 2 modifications: 1/ inversion using DLS to increase robustness with respect to bad matrix conditioning for the inversion and 2/ premultiplication by P_{J_O} in the term $P_{J_O}(J_C P_{J_O})^{+,DLS}$. This premultiplication enforces strict priority by projecting the errors induced by the approximation of the DLS inverse into the appropriate kernel (see appendix B.3.2).

- Control law B: Obstacle avoidance as 1^{st} task, trajectory tracking as 2^{nd} task (adapted from Sentis *et al.* (2005)):

$$\dot{\boldsymbol{q}} = J_C^{+,DLS}\dot{\boldsymbol{X}}_C + P_{J_C}(J_O P_{J_C})^{+,DLS}(\dot{\boldsymbol{X}}_O - J_O J_C^{+,DLS}\dot{\boldsymbol{X}}_C) \tag{4.2}$$

with P_{J_C} a projector on the kernel of J_C, as above. The "adaptation" from the control law of Sentis are the same than the previous ones done for Maciejewski's control law.

- Control law CCC: Constraint Compliant Control, as described in section 3.2.4.1.

$$\begin{aligned}\dot{\boldsymbol{q}} =& P_{J_c}(J_O P_{J_c})^{+,DLS}\dot{\boldsymbol{X}}_O \\ &+ P_{\left[\begin{smallmatrix}J_c\\J\end{smallmatrix}\right]}(J_{\bar{c}}\ P_{\left[\begin{smallmatrix}J_c\\J\end{smallmatrix}\right]})^{+,DLS}(\dot{\boldsymbol{X}}_{\bar{c}} - J_{\bar{c}}P_{J_c}(J_O P_{J_c})^{+,DLS}\dot{\boldsymbol{X}}_O)\end{aligned} \tag{4.3}$$

with J_c the Jacobian of constraints avoided passively, $J_{\bar{c}}$ the Jacobian of constraints avoided actively (complementary of J_c in J_C) and $P_{\left[\begin{smallmatrix}J_c\\J\end{smallmatrix}\right]}$ a projector on the kernel of the matrix containing the concatenation of J_c and J.

For each inspection mission, the manipulator must track a 3D trajectory (position without orientation); it is assumed that the effector (camera) would have the orientation DOFs needed to observe the points to be inspected. For the sake of simplicity, the results presented here do not integrate joint boundaries avoidance even though the proposed framework can deal with this type of constraint without any specific difficulty. These constraints are treated with the CCC in section 4.1.2.

The two missions can be described as follow:

- **Mission 1: Go around the wall by the left side.** The environment is barely constrained in that area, the manipulator tracks a trajectory (traj_1, in Fig. 4.1) of 330 points on 3.50 m, so an operational displacement of 11 mm is expected at each time step. The mission is *achievable*, i.e. the number of DOFs of the manipulator enables to fulfill the mission while avoiding the constraints.

- **Mission 2: Reach a point behind the wall.** The trajectory (traj_2, in Fig. 4.1) has 560 points, for a back and forth trajectory (to check that getting out of a very constrained configuration is not a problem). The total distance is 5.20 m long, so the expected operational displacement is 9 mm at each time step. The mission is not *achievable* as the manipulator is not long enough to reach the furthest point.

4.1.1.2 Implementation

The implementation is done in C++ within a velocity kinematic simulator coded during the thesis and named Nageval (Fig. 4.2). Nageval is linked to the following libraries and software:

- Kinematic and Dynamic Library (KDL), a library for modelling and computation of kinematic chains, from the OROCOS project (Bruyninckx 2001)

- CoDRoS, a distance computation engine based on Swift++ (Gamma Group, Ehmann *et al.* 2001)
- Graphic Display for Hilare Experiments (GDHE), a software for 3D visualization of robotics applications (Herrb 2010)

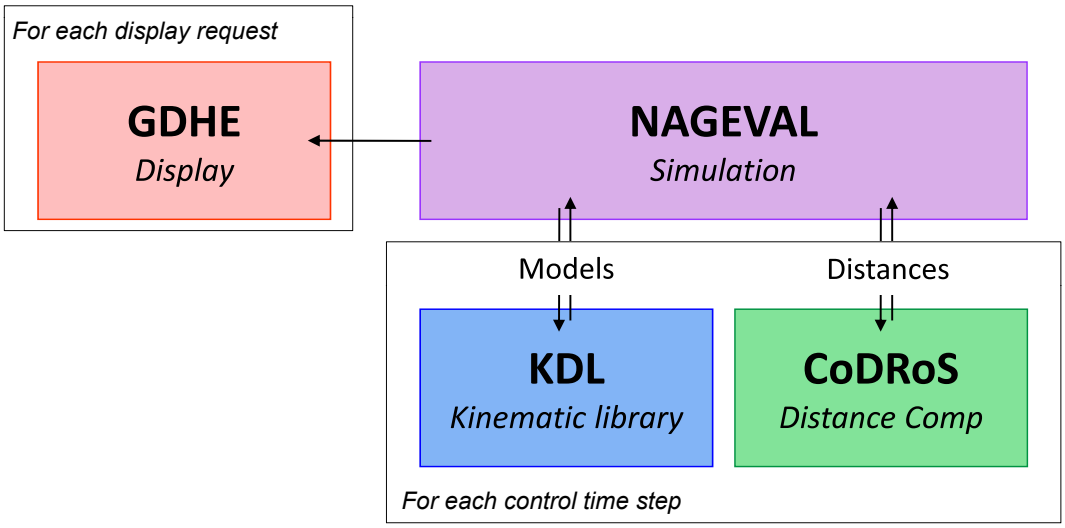

Figure 4.2: Environment of the Nageval simulator.

Regarding the controller, the following implementation elements are detailed:

- **Operational input**: the operational input $\dot{\boldsymbol{X}}_O$ is computed by the difference of the current effector position and the current desired effector position divided by the time step period δt (which value is transparent as the simulation is at the velocity kinematic level).
- **Pseudo-inversion**: the Damped Least Square pseudo-inversions are carried out with the damping factor λ chosen as 0.50 (see appendix B.2.2). This value for λ is a compromise between robustness with respect to singularities and operational error induced by the regularization (Hue *et al.* 2008).
- **Active avoidance**: for control laws A and B, the active avoidance threshold is fixed to $d_{ActAv} = 150$ mm, the gains are proportional (factor $\mu = 2.5 \quad 10^{-3}$) to the inverse of the distance to the constraint. For the CCC, the active avoidance threshold is fixed to $d_{ActAvCCC} = 40$ mm, the gains are the same than for control laws A and B but the maximum value of the avoidance magnitude $\dot{\boldsymbol{X}}_C$ is fixed to $\rho_{max} = 0.25$ which is equivalent to a distance of $\delta = 10$ mm between the manipulator closest point and the obstacle ($\rho_{max} = \frac{\delta}{d_{ActAvCCC}}$).
- **Distance computation**: as mentioned in section 3.2.4.1, given the unreliability of the avoidance techniques based on a unique point per segment constraint in discrete time, an envelope of 20 mm is added around the environment.
- **Joint velocity limit**: $\dot{\boldsymbol{q}}_{max}$ was fixed to 2 rad/s.

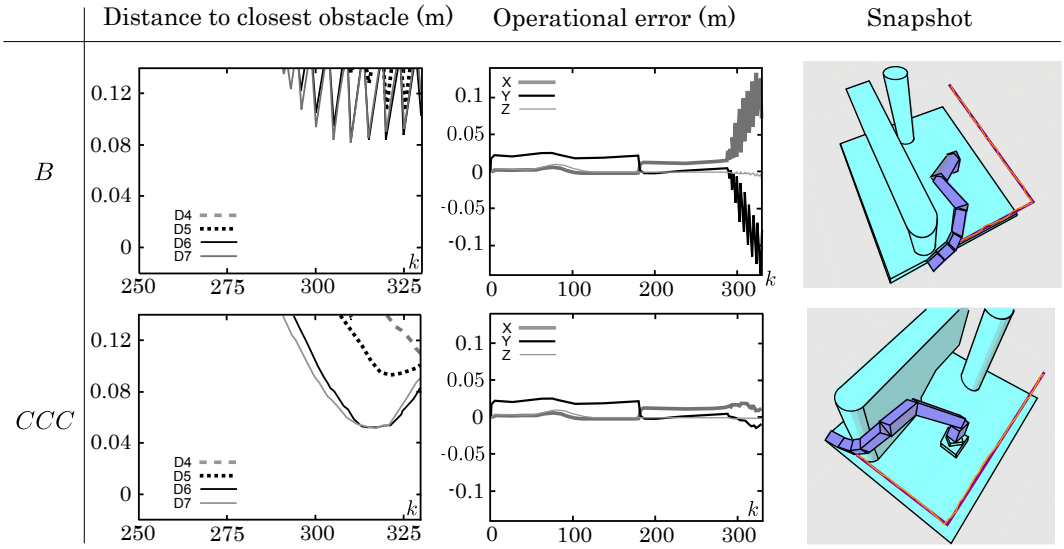

Figure 4.3: Mission 1 results. Graphs of line 1 are obtained with control B, line 2 with control CCC. The distances to the closest obstacle are drawn for the last segments (D4 to D7, see Fig. 4.1); the operational error concerns only position, as the trajectory does not involve orientation. The snapshots are views of the manipulator showing the position in the environment at the end of the simulation.

4.1.1.3 Results and Analysis

Figs. 4.3 and 4.4 present the results obtained on the 2 missions with control laws A, B and CCC. Some indications are given in the text about the computation times as a comparison basis, but they depend on the implementation and computing power. For all the control laws, the operational errors stabilize around 30 mm[1] in areas free of obstacles, what can then be considered as negligible error in the control laws comparisons. These errors are due to the linearization approximations induced by the resolution of the control problem at the velocity kinematic level.

- **Mission 1: Go around the wall by the left side.** As expected, the mission is achievable: the error can be maintained negligible without colliding (CCC results).
  - **Control law A.** As the behavior is identical to the CCC on underconstrained cases (see section 3.2.5), the behavior of the manipulator submitted to control law A is not represented on Fig. 4.3.
  - **Control law B.** The tracking is not optimal at the end where the effector gets close to the wall: oscillations are generated at the activation thresholds (observable on both graphs) and the operational error grows up to 131 mm. These oscillations can be well observed on the graph of distances to closest obstacle which focuses on the last time steps of the simulation. The period of those oscillations vary from 2 time steps to 6 time steps. These oscillations can be decomposed as follow:
    1. *Robot out of the activation area.* The obstacle is not taken into account, the robot tracks the trajectory which brings it close to the obstacle (1

[1]See graph 4.3 for example

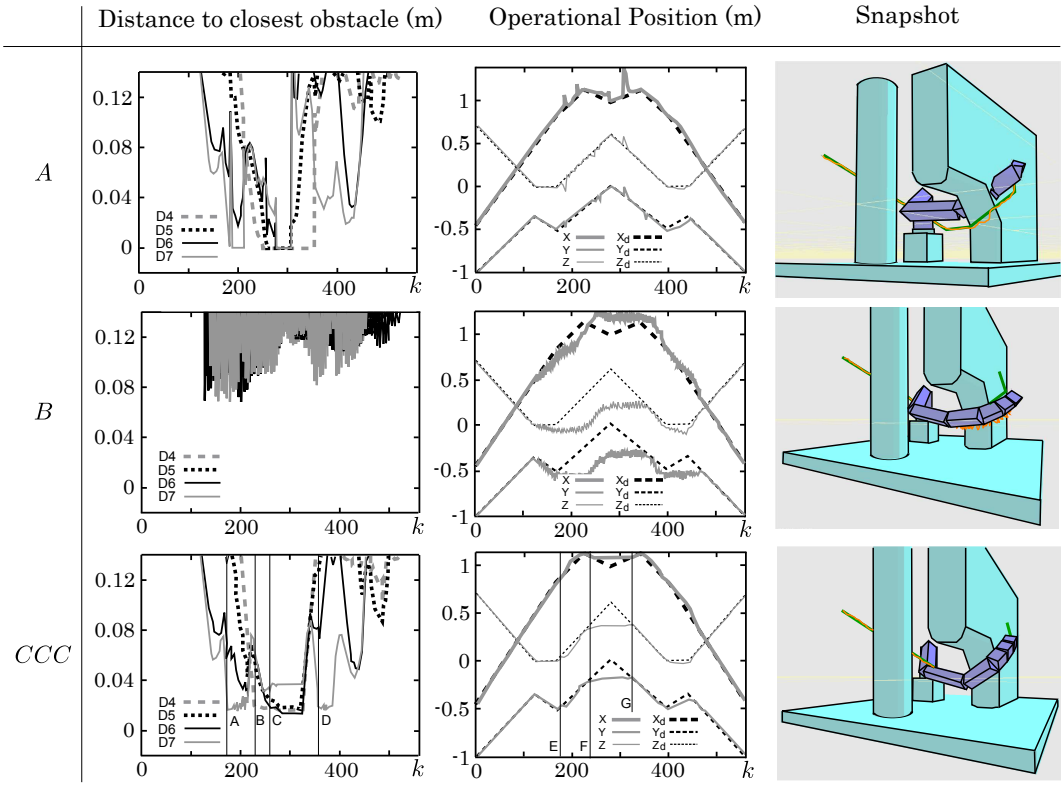

Figure 4.4: Mission 2 results. Graphs of line 1 are obtained with control A, line 2 with control B, line 3 with control CCC. The distances to the closest obstacle are drawn for the last segments (D4 to D7, see Fig. 4.1); the operational positions show the desired (X_d, Y_d, Z_d) and real (X, Y, Z) positions of the effector. The snapshots are views of the manipulator showing the position in the environment in the most constrained position (time step 300).

time step).

2. *Robot in the activation area.* The obstacle is taken into account in the control law at first priority level, so the robot avoids the obstacles and thus gets far from the trajectory tracking desired position. The number of time steps needed to get out of the activation area depends on the position within the area (1 to 5 time steps).

The obtained behavior is as expected the one of Fig. 3.4 (right). The safety of this approach is not proved; the behavior is not correct (oscillations); the solutions are not optimal (important errors).

– **Control law** CCC**.** The missions is completed successfully by the manipulator as the error remain below 30 mm, it does not collide and has a correct behavior. The distances to obstacle reach 50 mm but seem well balanced between joints 6 and 7. As there is no collision risk when tracking the path with active avoidance at a lower level, the behavior is identical to control law A (see Fig. 3.4 (left)). The robot tracks the trajectory and avoids obstacles if it does not impact the trajectory tracking; neither the passive avoidance, nor the scaling are used and the control problem is solved in a single iteration at

each time step. The behavior is correct and the solutions are optimal.

- **Mission 2: Reach a point behind the wall.**
  - **Control law** A**.** As shown on the snapshot, the robot collides considerably with the environment (D4: time steps 340 → 450; D5: 340 → 405; D6: 380 → 405; D7 190 → 210 and 380 → 405). The trajectory tracking being at the first priority level, the robot avoids the obstacles as long as this does not prevent from reaching the trajectory goal. As the mission is not achievable the robot collides. Each time a collision occurs, the velocity jumps because the active avoidance coefficient tends to infinity as it is proportional to the inverse of the distance to the obstacles. Except for this case, the trajectory tracking is quite perfect. For implementation reasons, the distance between the environment and the segments is not considered anymore when the robot is in collision; that is why there is only a jump and no permanent infinite velocities when in collision.
  - **Control law** B**.** As for mission 1, the robot behavior is correct until it reaches the obstacle activation thresholds. Then, oscillations occur and maintain the robot far from the environment and the trajectory target. Even if the robot does not collide in this case, safety is not guaranteed as it relies on the activation threshold width.
  - **Control law** CCC**.** First, the robot does not collide (and the distance remains superior to the envelope of 20 mm), the trajectory tracking presents no oscillation and the most constrained situation (snapshot) shows that the robot is extended without reaching the trajectory tracking target (mission not achievable); the come back is done successfully (robot not stuck to the constraints). When the manipulator comes close to the environment (envelope distance: 20 mm), the passive avoidance clamps the directions to the obstacles (point A, B, C and D on the first graph) and the concerned segments moves along the orthogonal directions. When the manipulator seems completely constrained (point E on the second column graph), the scaling step enables little displacements to track as much as possible the trajectory. The robot is safe, the behavior is correct ad the tracking is close to optimality (see section 3.2.5). The computation time is multiplied by a factor of 6.7 times with respect to the ones obtained with control law A on mission 1.

4.1.1.4 CCC results conclusion

The results presented here illustrate the ability of the CCC algorithm to solve hierarchical multi-objective control problems while satisfying any number of constraints allowing instantaneous stops: for example obstacles, joint boundaries and joints velocity limits. The passive avoidance principle and the solutions scaling enable to overcome the drawbacks of active avoidance at the top (optimality loss, oscillations) or at the bottom (constraints violation, infinite terms) of the hierarchy, while ensuring a computation time low enough to consider its use in real time on classical manipulators.

The CCC performances can be compared to the one obtained with convex optimization algorithms (even if the latter cannot ensure strict priorities between the hierarchy levels without resorting to sequential programs, see section 3.1.3.1). As an example, the

algorithm QuadProg++[2] satisfies the constraints with a maximum computation time of 3 times the one of control law A. As strict priority between multiple objectives cannot be ensured in a simple framework, the absence of active avoidance in the implemented control law contributes to the algorithm tendency to run along the constraints. It makes the manipulator not able to get away of the most constrained configurations: as an example, it does not track the second part of the trajectory (it remains stopped in a constrained configuration in the most extended position).

However, the CCC presented in this section cannot deal properly with joint acceleration constraints as it relies on the fact that the current configuration is always admissible - belongs to E_C -. The displaced configuration principle presented hereafter enables the CCC to consider all kinds of constraints.

4.1.2 Displaced configuration control results

The simulations presented hereafter illustrate the behavior of control laws using displaced configurations. The first simulation is a simple example of a 3R planar manipulator submitted to constraints of joint position, velocity and acceleration limits. The second simulation involves a 7R manipulator in 3D which handles the same constraints plus obstacles.

4.1.2.1 3R planar manipulator: simulation presentation

As an exception, this simulation is carried out through arboris[3], a dynamic simulator coded in python and used for example in Salini *et al.* (2011).

This simulation involves a 3R planar manipulator in charge of tracking a trajectory with a single iteration constraint compliant control law (as described in section 3.3.3). The aim of the simulation is to check on a simple example that the manipulator tracks properly the trajectory while satisfying the constraints. The considered constraints are:

- Joint position limits - control constraints Eqs. (2.29) and (2.30);
- Joint velocity limits - control constraints Eqs. (2.23) and (2.24);
- Joint acceleration limits - control constraints Eqs. (2.26) and (2.27).

As there is no obstacle to avoid, Alternative Safe Behaviors (section 2.4.5) are not required as the control constraints are made compatible (see section 2.4.3, case study 2). The features of the manipulator are presented on table 4.1. The manipulator is presented on Fig. 4.5.

The tracked trajectory consists of a 2D point moving toward the basis of the manipulator. The mean distance between two points of the trajectory is around 3 mm. As the robot has 3 DOFs, it is redundant with respect to the trajectory tracking task (J_1, $\dot{\boldsymbol{X}}_1$); a second task (J_2, $\dot{\boldsymbol{X}}_2$) is arbitrarily applied on the third joint (blue arrow on Fig. 4.5) between $t = 0.6\ s$ and $t = 1\ s$ to check that a sudden Jacobian rank change does not lead to a velocity discontinuity (Mansard *et al.* 2009a); finally, at $t = 3s$, a

[2]http://sourceforge.net/projects/quadprog/ This algorithm is a quadratic program solver relying on the active set dual method of Goldfarb and Idnani (1983)

[3]Arboris-Python, by S. Barthélemy, J. Salini and A. Micaelli. https://github.com/salini/arboris-python

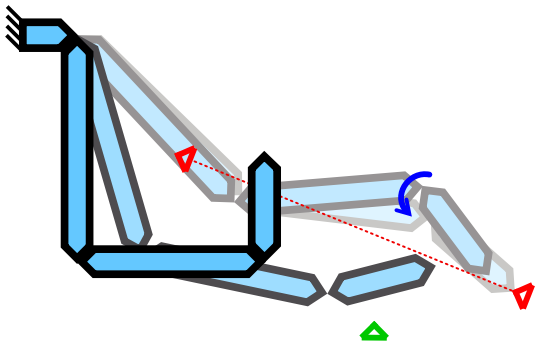

Figure 4.5: The 3R planar manipulator motion sequence.

Table 4.1: 3R Characteristics

	1	2	3
Segment Length (m)	0.5	0.4	0.2
Joint limit (rad)	$\pm\pi/2$	$\pm\pi/2$	$\pm\pi/2$
Vel limit (rad/s)	± 1	± 1	± 1.5
Acc limit (rad/s^2)	± 5	± 5	± 5

new operational objective is given (green triangle on Fig. 4.5) to check that the robot is able to move away from its constraints.

The retained displaced configuration is at the center of E_C (Eq. (3.32)). The control law is recalled here (Eq. (3.38))

$$\dot{\boldsymbol{q}}_{\boldsymbol{dp}} = J_1^{\#,DLS}\dot{\boldsymbol{X}}_{1,dp} + P_{J_1}(J_2P_{J_1})^{\#,DLS}(\dot{\boldsymbol{X}}_{2,dp} - J_2J_1^{\#,DLS}\dot{\boldsymbol{X}}_{1,dp}) \tag{4.4}$$

where the pseudo-inversion are approximated (DLS) with $\lambda = 0.5$ and weighted by the joint space admissible widths (see section 3.3.2.5). When the second task is active (between $t = 0.6\ s$ and $t = 1\ s$), $J_2 = [0 \quad 0 \quad 1]$ and $\dot{X}_2 = \alpha(q_{des} - q_3)$ where $\alpha = 30\ m.s^{-1}.rad^{-1}$ and $q_{des} = 0.5\ rad$.

4.1.2.2 3R planar manipulator: results

The simulation sequence is presented on Fig. 4.5. The graphs of positions, velocities and accelerations for each joint are presented on Fig. 4.6. Every constraint is always satisfied (grey horizontal lines). As expected, the joint positions and velocities are smooth, and the velocity slopes are bounded. When some segments of the manipulator reach their joint position limits, the manipulator continues to move even if the control problem is solved in a single iteration.

- At time step A, the objective of second priority is added, which is supposed to cause a discontinuity in the joint velocities with usual control methods. In the presented case, the velocity slope changes but the accelerations remain between their limits. The second objective is left at time step B, and the same phenomenon occurs.

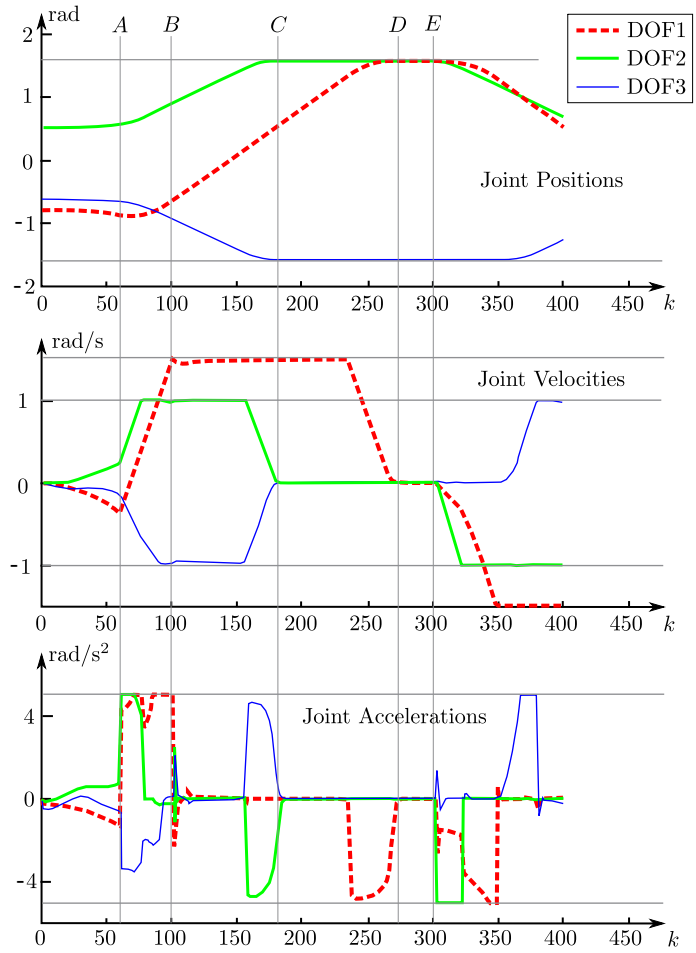

Figure 4.6: Joints position, velocity and acceleration with respect to time steps.

- At time step C, joints 1 and 2 reach their position limits simultaneously. These joints were decelerating since time step 150 to maintain joints position and acceleration limits compatible, according to the expected behavior established in case study 2 of section 2.4.3. Then, even if joint 1 and 2 cannot contribute to the motion, joint 3 continues to track the trajectory at best until its position limit (time step D).

- Finally, at time step E, the desired operational direction changes (green point on Fig. 4.5) and the manipulator gets away from its constrained configuration.

Between time steps C and D, the use of a displaced configuration in the control is required to continue the motion while joint 1 and 2 are on their position limit. Actually, without it the scaling would prevent any motion of the manipulator (see Fig. 3.14). The use of the displaced configuration at the center of E_C moves $\boldsymbol{q}_b$ away from the position limits, so the motion is not scaled by 0 and some joint motions are possible.

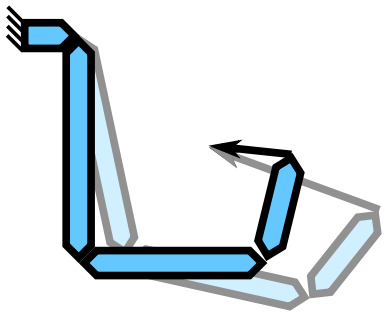

Figure 4.7: Lack of optimality of the displaced configuration technique. In grey, the displaced configuration. The arrows are the operational input. Without the if loop used in the proposed simulation, the robot configuration might have converged toward the black configuration: conversely to joint 1 and joint 2, joint 3 has not completely reached its limit. The homogeneous scaling makes the motion from the displaced configuration not optimal (see Fig. 3.16). In the presented simulation, the fold up configuration is reached thanks to the full deceleration enforced by the constraints.

However, as mentioned in section 3.3.3 and as showed on Fig. 4.7, the robot submitted to a displaced configuration based control tends to converge toward a non optimal configuration. Actually, this is not the case in the presented simulation thanks to the way the joint position reaches its limit. During the deceleration, the constraints compatibility enforces a full deceleration until the joint position limit. As shown in Fig. 4.7, the fold up configuration is not stable: the displaced configuration is *behind* the robot with respect to the point to reach, as the centre of E_C gets away from the joint position limit. However, as suggested in section 3.3.3, this case is treated by an if loop to maintain the robot in the best configuration.

This method may turn out insufficient to carry out properly the tasks. This can be the case when the robot moves slowly toward one the fold up configuration. In these cases, any approach iterating on the passive avoidance of constraints (as the clamping method in Baerlocher *et al.* (2004) or the CCC presented in section 3.2) can be used.

4.1.2.3 7DOFs manipulator: simulation presentation

This simulation is an extension of the one presented above: it involves a 3D manipulator in charge of tracking a trajectory in presence of obstacles. The simulator used is Nageval (velocity kinematics). The aim of this simulation is to illustrate the behavior of the CCC law with a displaced configuration located at the maximum deceleration point (cf. Eq. (3.33)) in a complex case. The considered constraints are:

- Joint position limits - control constraints Eqs. (2.29) and (2.30);
- Joint velocity limits - control constraints Eqs. (2.23) and (2.24);
- Joint acceleration limits - control constraints Eqs. (2.26) and (2.27);
- Obstacles - control constraints (2.25).

As for the previous simulation, the presence of joint acceleration limits imposes to use the constraints function of Eqs. (2.29) and (2.30) rather than Eqs. (2.21) and (2.22) for joint position limits. The simultaneous presence of joint acceleration limits and obstacles imposes to use Alternative Safe Behaviors (section 2.4.5). In this particular case, the ASB retained is the maximum joint deceleration (section 2.4.5.1).

The manipulator has 7 rotational DOFS, it is the same than the one used in the simulations of section 4.1.1 but in a different environment (Fig. 4.8). The joint positions of the robot are limited to $\pm\frac{\pi}{2}$ rad , the velocities ± 1.5 $rad.s^{-1}$ and the accelerations ± 10 $rad.s^{-2}$. The simulator used is Nageval.

The tracked trajectory is composed of 3D points (position without orientation) which by-pass the wall (Fig. 4.8). The mean distance between two points of the trajectory is 8 mm. The mission is not achievable, *i.e.* the manipulator is geometrically not able to track the trajectory perfectly.

The retained displaced configuration is the maximal deceleration configuration (cf. Eq. (3.33)), which is efficient on smooth operational tasks (see section 3.3.1.2). The environment envelope e is 10 mm. The control law used is the one of Eq. (4.3) recalled hereafter with 1/ pseudo-inversions weighted by the joint interval width (see section

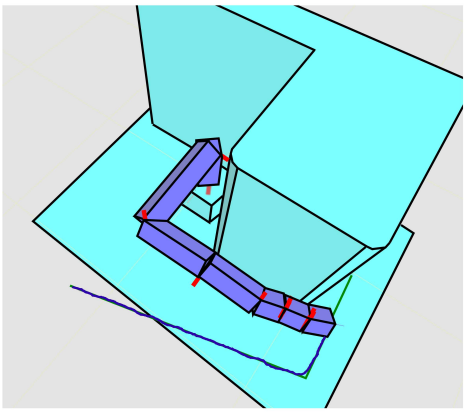

Figure 4.8: 7 DOFs manipulator in the last configuration of the simulation. The joint axes are represented in red.

3.3.2.5), 2/ the trajectory tracking as unique objective in $(J_O, \dot{\boldsymbol{X}}_{O,dp})$ and 3/ the constraints terms $((J_c, \mathbf{0}), (J_{\bar{c}}, \dot{\boldsymbol{X}}_{\bar{c},dp}))$.

$$\begin{aligned}\dot{\boldsymbol{q}} =& P_{J_c}(J_O P_{J_c})^{\#,DLS}\dot{\boldsymbol{X}}_{O,dp} \\ &+ P_{\left[\begin{smallmatrix}J_c\\ J\end{smallmatrix}\right]}(J_{\bar{c}}\ P_{\left[\begin{smallmatrix}J_c\\ J\end{smallmatrix}\right]})^{\#,DLS}(\dot{\boldsymbol{X}}_{\bar{c},dp} - J_{\bar{c}}P_{J_c}(J_O P_{J_c})^{\#,DLS}\dot{\boldsymbol{X}}_{O,dp})\end{aligned} \tag{4.5}$$

The time step increment is $\delta t = 0.01\ s$.

4.1.2.4 7DOFs manipulator: results

The graphs of joints positions, velocities and accelerations of DOFs 1, 4 and 7 as well as the distances to the nearest points of the environment are presented on Fig. 4.9. As in the previous experiment, the positions and the velocities are smooth, and every constraint is always satisfied. At time step 95, the first DOF reaches its position limits, which provokes an acceleration of the other DOFs to maintain the desired operational velocity. At time step 135, the robot begins its deceleration close to the obstacle; the ASB provokes a deceleration of the whole robot as the prediction loop ϕ_{ASB1} detected a collision with the envelope. When all the DOFs are stopped (time step 175), DOF7 has a margin to continue to minimize the tracking error until it reaches the envelope of the environment. As expected in section 2.4.5.2, the behavior of DOF7 presents some oscillations before the final deceleration (time steps 185 $\rightarrow$ 195), showing the limits of the maximum deceleration ASB.

4.1.3 Control problem resolution results conclusion

In this section, two sets of simulations are presented:

- The first one uses the CCC in a context involving joint position, velocity limits and obstacles. The simulation illustrates that the control law has a behavior similar to state-of-the-art control laws in underconstrained cases. In over-constrained cases, it fulfills the multiple objectives while satisfying the constraints without the drawbacks encountered with state of the art control law (high lack of optimality and oscillations). These simulations validate this approach for an implementation in the multi-objective evolutionary design process (section 4.2).

- The second set of simulations involves the same constraints and joint acceleration limits. These simulations illustrate the ability of the displaced configuration control to fulfill the objectives while satisfying the constraints, either with single iteration control law or with the CCC.

4.2 Evolutionary design

This section is dedicated to the design of a manipulator in charge of the maintenance operations (cleaning, inspection) in the Tunnel Boring Machine excavation room of the TELEMACH project. The environment is cluttered and the mission requires high reachability skills.

The kinematic design of robotic manipulators is often seen as one among numerous applications of engineering design. When robots are uniquely used as chain manufacturing tools, this vision is appropriate and classical design methods (Pahl *et al.* 2007)

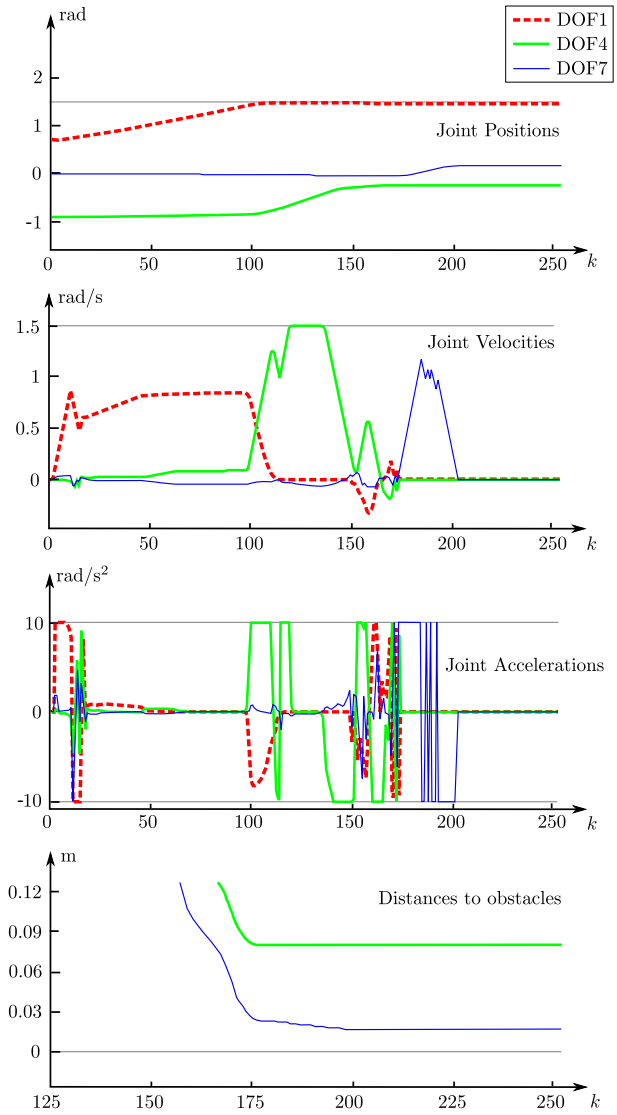

Figure 4.9: Joints positions, velocities and accelerations with respect to time steps along the simulation. The last graph is the distance to environments. It focuses on the second half of the simulation (time step 125 → 250). The constraints are represented by horizontal grey lines; all the variables remain between their constraints.

turn out to be rather efficient especially when combined with dedicated CAD and PLM[4] tools, allowing to numerically evaluate potential solutions.

The complexity of TBM environment and the tasks to be accomplished induce a high number of potential design solutions among which the best ones may, given their originality with respect to usual design problems solutions, probably not arise using classical design methods. The use of dedicated CAD tools may help to numerically discard some of the potential solutions, but checking each robot candidate with respect to a representative subset of tasks and environments still remains a complex and time consuming work.

Recently, the enormously increasing popularity of optimization methods led to their effective application in robotic design. Amidst several works presented for optimal designs of fundamental robots Kim *et al.* (1992), Chen *et al.* (1995), Lee *et al.* (2004). Ceccaralli and Lanni (2004) involved two conflicting design objectives: maximum work volume and minimum link lengths. The problem is converted into a single objective problem through a weighted sum approach and solved using Sequential Quadratic Programming (SQP). More recently, a multi-objective optimal design algorithm for 6-dof PUMA robots was discussed by Carbone *et al.* (2007). Though, these works introduce the simultaneous fulfillment of multiple design criteria through optimization techniques, the methods used for solving the problems fall short in providing multiple optimal solutions. Besides, the complexity associated with cluttered environments and larger number of degrees of freedom is left unaddressed. A very recent work presented by Singla *et al.* (2010) utilizes another classical optimization technique, called Augmented Lagrangian approach, for designing redundant manipulators working in constrained workcells. Although the paper deals with practical trajectories in constrained environments, it possesses the limitation of fixing the number of degrees of freedom *a priori*.

Instead, it is proposed to follow an approach where the design process is considered as a multi-objective optimization problem: tasks and constraints are formulated in terms of functions to optimize and constraints to satisfy. Such a formulation allows the automation of the design process in the preliminary phase. Given a family of automatically obtained solutions, the so-called classical design methods can then be used to converge towards a practical solution.

The presence of multiple objectives in a problem gives rise to a set of optimal solutions, instead of a single optimal solution. This set of solutions is known as the set of Pareto-optimal solutions and rely on the notion of *Pareto-dominance* (Deb *et al.* 2001) to treat simultaneously and independently each performance indicator.

In a typical minimization problem where the fitness (evaluation of the individuals) f is composed of n functions f_i $(1 \leq i \leq n)$, a solution x is dominating an other solution x' if

$$\exists\ i \text{ such as } f_i(x) < f_i(x') \tag{4.6}$$

and

$$\forall j \neq i, f_j(x) \leq f_j(x') \tag{4.7}$$

Based on this principle, the solution of the multi-objective optimization is the set of non-dominated solutions to the problem (Pareto-optimal solutions). In the absence of any further information, one of these Pareto-optimal solutions cannot be said to be better than the other.

[4]Product Lifecycle Management

In a first section, the literature review exposes the principles of Multi-objective optimization through Genetic Algorithms (GAs). The choices made for each step of the design process are detailed. The third section focuses on the control law used in the simulations to evaluate the robots. Finally, some results are exposed and discussed.

4.2.1 Genetic Algorithm and architecture adopted

4.2.1.1 Context

The work is oriented toward task based design through the evaluation step of a GA (see Fig. 4.10).

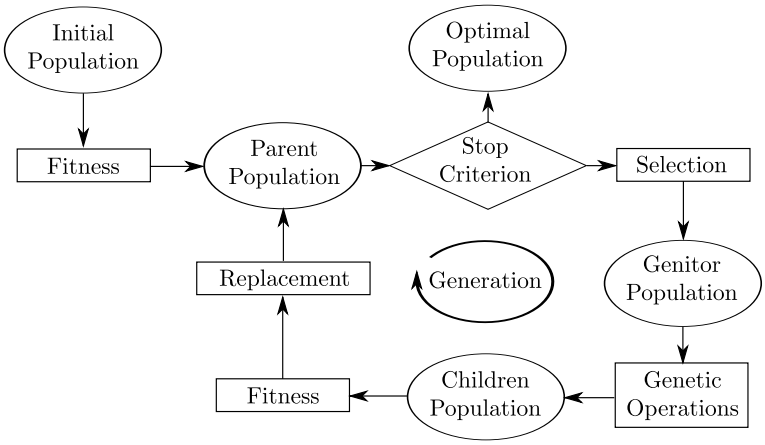

Figure 4.10: Genetic Algorithm general scheme. The initial population is evaluated through the fitness function, then the generation loop begins. A stop criterion is applied to the resulting parent population. If the criterion is not verified, the selection process retains the individuals (genitor population) on which to carry out the genetic operations (most of the times *crossover* and *mutations* on the genotypes representing the individuals). The resulting population is then evaluated, and the replacement produces the new parent population.

In this framework, a huge number of a individuals (robots) are evaluated thanks to the *ad hoc* velocity kinematic simulator Nageval (see section 4.1.1.2), which is linked to Sferes$_{v2}$, a framework for evolutionary computation experiments (Mouret *et al.* 2010) (see Fig. 4.11).

4.2.1.2 Genetic Algorithms literature review

Genetic Algorithms are optimization processes inspired from natural evolution. The first introductions of GAs come from the books of Holland (1992) and Goldberg (1989). At that time, GAs were used in mono-objective optimization framework. GAs are a part of Evolutionary Algorithms (EAs), which gathers all the algorithms resorting to techniques inspired from natural evolution.

One of the first robot design problems using Evolutionary Algorithms (EAs) was carried out by Sims (1994), generating creatures competing in walking, jumping, swimming, etc. The concept of dominance (Eqs. (4.6), (4.7)) introduced by Pareto in 1906 was then exploited to open the way for Multi-Objective Evolutionary Algorithms (MOEAs).

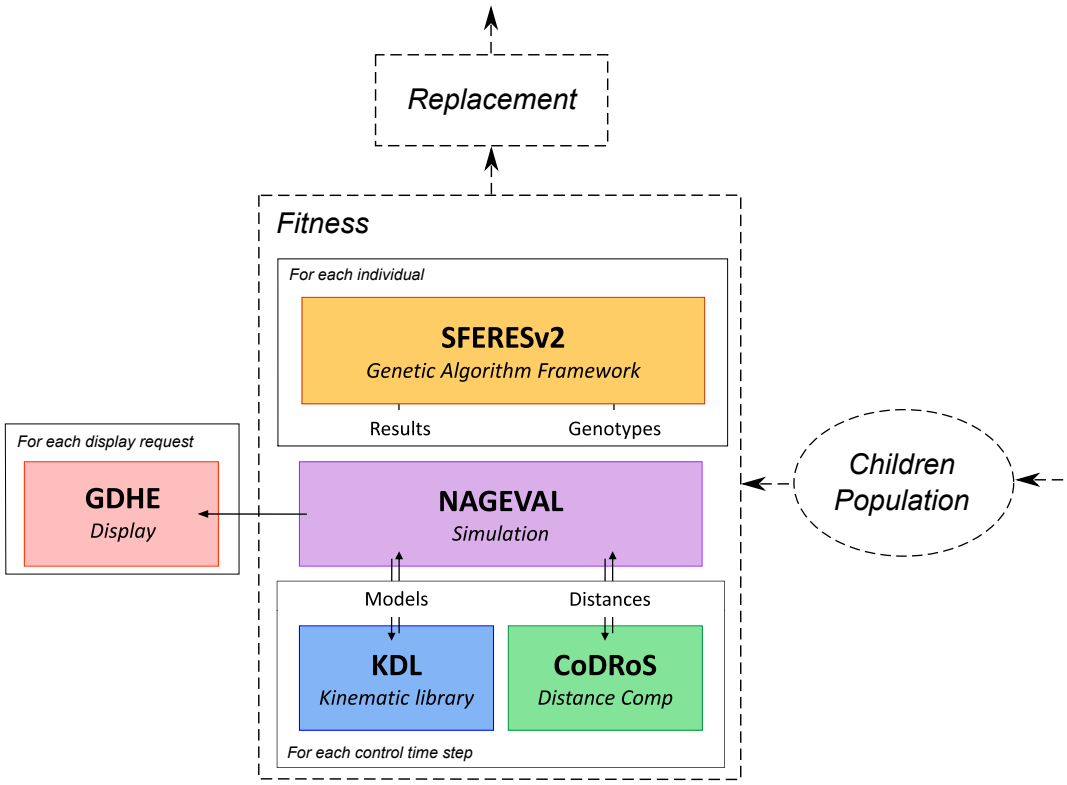

Figure 4.11: Architecture of the Evolutionary Design software. Each individual of the Children Population, expressed by a genotype, is evaluated through the fitness thanks to the Nageval simulator which returns the Results, *i.e.* the scores of the individuals.

Since 1990, a high number of MOEAs have been proposed (Fonseca *et al.* 1993, Horn *et al.* 1994, Srinivas *et al.* 1994, Zitzler *et al.* 1998, Deb 2001, 2002). GAs are now widely used in robotics as they are very well adapted for optimization over vast, non continuous search space. Applications fields spread from the whole system structure design ((Chocron *et al.* 1997, Leger 1999), Gallant *et al.* 2002) to robots reconfiguration (Han *et al.* 1997), controller design (Cheng *et al.* 1997), and in various domain such as cooperative robotics (Sotzing *et al.* 2005) and mini-invasive surgery (Salle *et al.* 2004).

4.2.2 Design formulation and resolution method

In this section, the stakes and the choices made for each step of the design process are exposed and discussed.

4.2.2.1 Genetic Algorithm

The efforts made to approach the Pareto-optimal front involve two (possibly conflicting) objectives:

- **Convergence**: minimizing the distance between the final Pareto front and the optimal front;
- **Diversity**: maximizing the difference in the generated solutions in terms of objectives or parameter values.

To consider both items, the commonly used technique of Nondominated Sorting Genetic Algorithm II (NSGA-II) (Deb *et al.* 2002) is considered suitable for the considered design problem. This technique possesses the features of *elitism* and *parameter-less selection*. Elitism is the process of selecting the best solutions out of the combined population of parent and child generations and, therefore, avoiding the elimination of any good solution. For a problem with the population size as N, NSGA-II works on $2N$ solutions at each iteration. These solutions are sorted with respect to their non-domination and are arranged into different Pareto optimal fronts. This is termed as non-dominated sorting. Within a given Pareto front, all candidates are equally good; to obtain the N individuals (genitor population), the best Pareto front are integrally kept until the number of individuals gets superior to N. The extra individuals are removed according to the crowding distance preference: the most isolated individuals on the objective space are preferred to support exploration. This helps maintaining some significant diversity in the resulting solution, by selecting widely spread population.

It is important to notice that the complexity of the problem resolution (convergence/diversity) depends not only on the size of the search space, but also on the number and on the nature of the retained indicators. The number of indicators increases exponentially the dimension of the pareto front space. If these indicators are highly non linear with respect to the retained representation of the search space, the algorithm is ill-conditioned. Actually, both the dimension and the shape of the indicators space are critical.

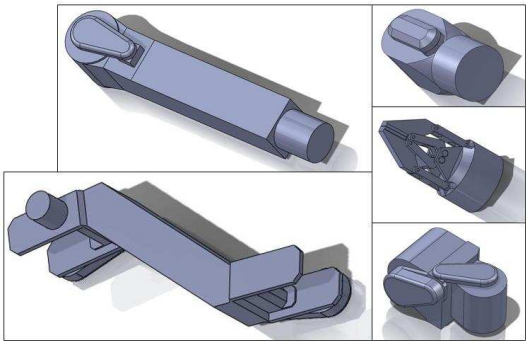

Figure 4.12: Example of robot segments of the Maestro manipulator from which the genome is inspired.

4.2.2.2 Genotype

The design process focuses exclusively on the robot morphology using elementary segments such as the ones shown in Fig. 4.12. In that framework, each robot is described as a concatenation of segments. A segment is composed of a link having a joint (rotational) or not. Two frames are associated to each elementary segment. The first one represents the three possible joint axes: every link is oriented along its z axis. The second one represents the three possible orientations of the next segment. According to this description, there are 11 elementary segments (see Fig. 4.13):

- 3 with a rotational joint about the x axis, the following segment being oriented along x, y or z (called rxx,rxy, and rxz respectively)

Table 4.2: Genome

Gene **ABC**	**AB**: Joint type
	C: length (m)

Joint number - **AB**	10	11	12	13	14	15
Joint type	No	rxx	rxy	rxz	ryx	ryy
Joint number - **AB**	16	17	18	19	20	21
Joint type	ryz	rzx	rzz	ex	ey	ez

Length number - **C**	0	1	2	3	4
segment length (m)	0.05	0.15	0.25	0.35	0.45
Length number - **C**	5	6	7	8	9
segment length (m)	0.55	0.65	0.75	0.85	0.95

- 3 with a rotational joint about the y axis (ryx, ryy and ryz)
- 2 with a rotational joint about the z axis (rzx and rzz)
- 3 segments without joint (ex, ey and ez)

rzy is not mentioned as it is the same as rzx rotated by $\frac{\pi}{2}$ rads around z axis. In addition we define 10 possible lengths for the segments between 0.05 m and 1.05 m. The association table is presented in Table. 4.2. As an example, a portion of a robot is represented on Fig. 4.14.

Each robot is defined by a chromosome of 16 genes, each one representing a segment or not: the genes from 100 to 109 do not match anything (segment "None") which is consistent with the fact that we do not want every robot to have 16 DOFs. When a fixed segment appears in the genotype of an individual (gene 190 to 219), a segment combination is done, thus offering the possibility to get segments which orientation differs from the x, y and z axes (Fig. 4.15).

As a conclusion, the retained genome (representation) has the following features (Palmer *et al.* 1995):

1. The representation is able to represent all possible robots;
2. The representation does not encode unfeasible solutions;
3. The representation is redundant, as several genotypes can encode the same individual.

Features 1 and 2 are part of the recommendation of Palmer; other considerations such as *locality* (small changes in the genotype should result in small changes in the phenotype) and *balance* (all possible individuals are equally represented in the set of all possible genotypic individuals) have been neglected to keep the genome expression simple. Feature 3 is largely induced by genes 100 to 109 ($\rightarrow$ no segment) and seems useful as it eases the access to simple morphologies. Redundancy is often seen as profitable as

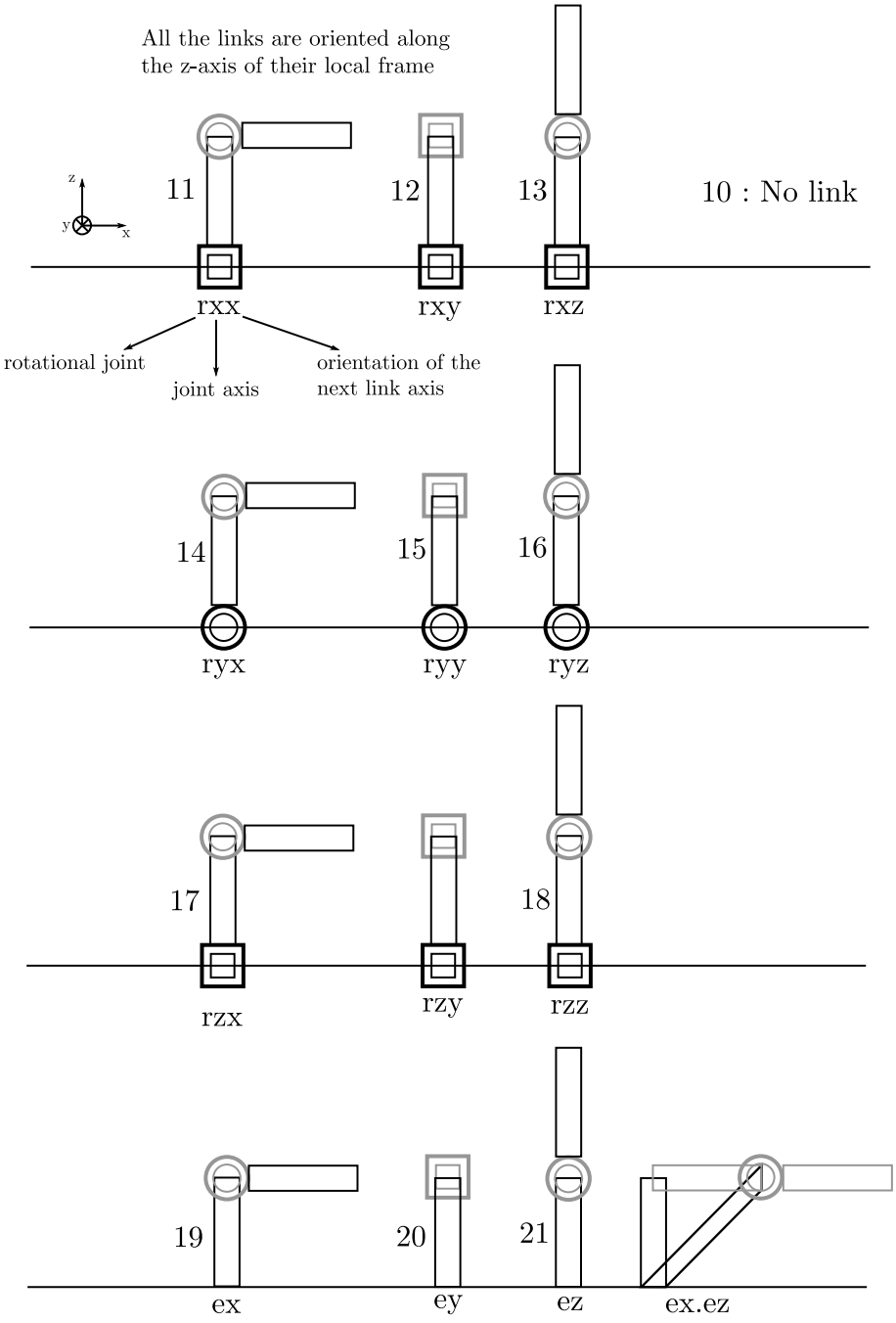

Figure 4.13: Scheme of the GA representation (genotype).

it increases the evolvability of the systems through neutral networks (set of genotypes connected by single-point mutations which map to the same individual). However, it increases the randomization of the search. Recent discussions about genetic algorithms representations can be found in Rothlauf (2006).

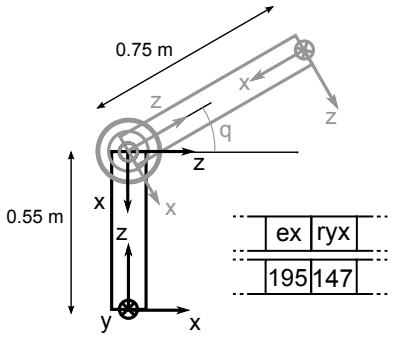

Figure 4.14: Portion of robot. q is the rotational joint angle.

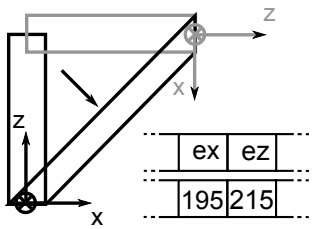

Figure 4.15: Combination of 2 segments.

4.2.2.3 Missions

The aim of the fitness function is to qualify the ability of a robot to carry out a maintenance mission in the TBM (Figs. 4.16, 4.17). So, a relevant trajectory has been defined in the simulation environment and the fitness function consists of a trajectory tracking. The trajectory is composed of 360 3D points without orientation, it is assumed that the end-effector held by the manipulator has the needed DOFs to perform the operations properly once it is at the correct operational position. As every robot has to track the same trajectory, an automatic presimulation executed for each robot is in charge of bringing the manipulator end effector near the trajectory first point. This presimulation remains an open problem as the interference between the robot and the environment is resolved by the elimination of the individual, which is not optimal.

The simulation is done at the velocity kinematic level, as dynamics is time consuming and offers poor advantages in this particular case: the fitness evaluates the geometry of the robot (is the robot able to reach the desired points without colliding with the environment) and not the ability of the controller to track the trajectory with a given dynamics.

4.2.2.4 Genetic parameters

The genetic operators (crossover and mutations) are regularly used at various rates, and their tuning are sometimes included in the optimization process (Srinivas *et al.* 1994). The crossover probability influences the injection of new solutions in the problem, which must not be too high to be exploited properly by the selection step. The mutation is often considered as a secondary operator to restore the genetic material; it should not be too high to avoid turning the GA into a random search algorithm. In our case the values have been taken as

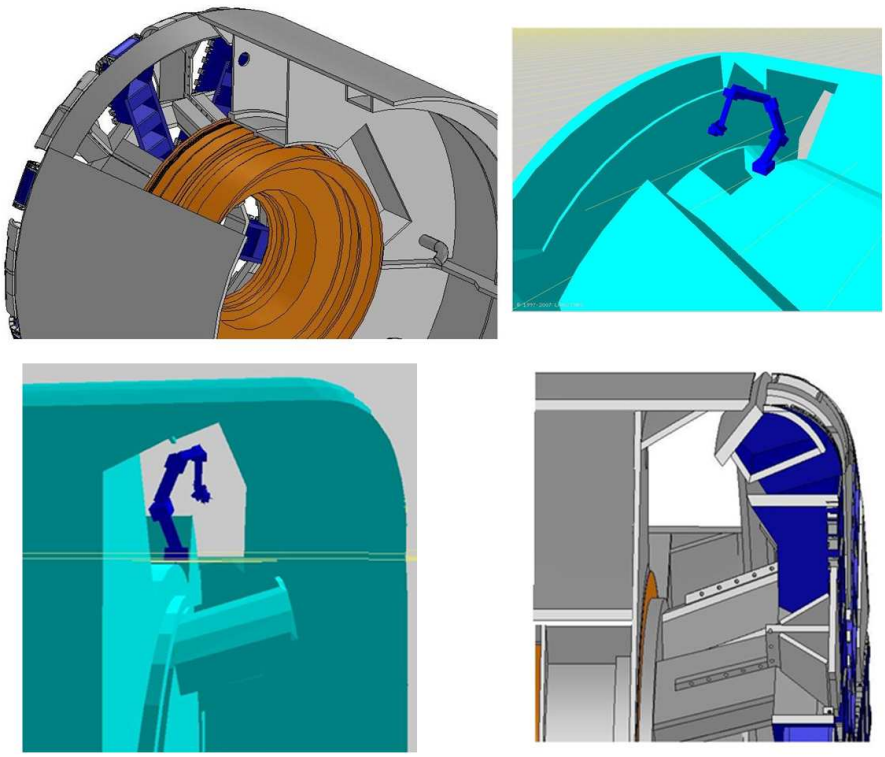

Figure 4.16: Comparative views of the TBM CAD and the simulation environment.

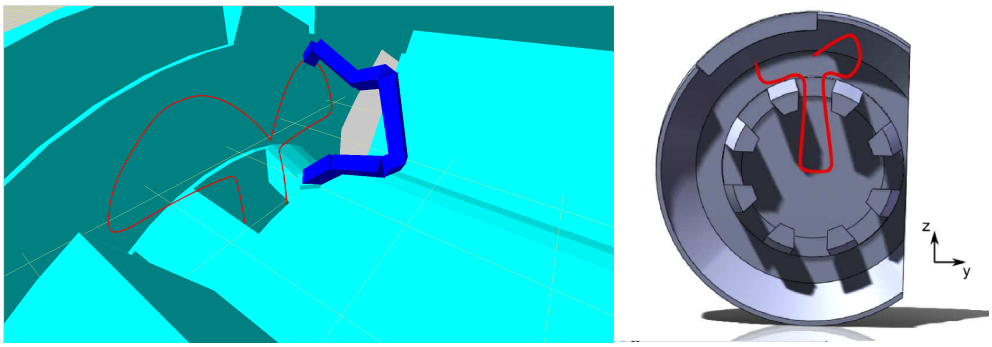

Figure 4.17: 7 DOFs Robot and trajectory example in the TBM environment.

- Crossover rate: 13%
- Mutation rate: 10%

The population size and the number of generations is limited by the design process time consumption. As one evaluation takes around 300 ms, a night-fitted design process is obtained with

- Generations: 500
- Individuals: 150

4.2.2.5 Indicators

The indicators are the criteria on which the robot is evaluated through the fitness function. Independently from the search space dimension (defined by the genotype), the number of indicators has a direct impact on the behavior of the problem resolution. For a M-criteria problem, a rough approximation could assimilate the Pareto front to an hypersurface of dimension $(M-1)$; when the dimension increases, maintaining an equal resolution quality for the Pareto front description requires a number of individual exponential in M. However, the relations between the indicators impact the Pareto front shape and thus the complexity of the problem; these relations can be (Purshouse *et al.* 2003):

- **Independence** The relation of independence qualifies the absence of influence between indicators. For example, maximizing the number of DOFs while maximizing the length of each segment. In that case, the problem can be divided into subproblems and the criteria can in theory be optimised separately from each other.
- **Conflict** The relation of conflict qualifies the impossibility to satisfy both of them simultaneously. For example, minimizing the size of a robot while maximizing its workspace. This is the usual reason for resorting to multi-objective optimization, as the Pareto front contains all the compromises between these indicators. This relation has the worst impact on the algorithm convergence.
- **Harmony** The relation of harmony qualifies the fact that the variation of one indicator is likely to provoke the same variation on the other indicator. For example, maximizing the size of a robot is in harmony with maximizing its workspace. This has only a small effect on the algorithm convergence.

In practice, relations between indicators are rarely pure and often change according to the search space area. A general recommendation in Genetic Algorithm based optimization is to limit the number of conflicting criteria to 3.

The indicators retained in our problem are voluntarily simple and of a single dimension (no weighted sums representing *a priori* tradeoffs between different variables). The trajectory tracking quality but also intrinsic parameters, such as the number of DOFs, are evaluated. All the retained indicators are to be minimized:

- **Maximum linear error along the trajectory tracking.** There are no strategic points on which to compute the error with a higher weight with respect to others: the current design being a preliminary design, the trajectory should be equally tracked;
- **Number of DOFs.** The number of DOFs is a technological difficulty (manufacturing, energy, control), even if the degree of redundancy usually increases the reachability skills in a cluttered environment;
- **Robot total length.** The shortest robots able to perform the trajectory tracking in a cluttered environment have usually better adaptability to other tasks. In addition, their energetic rates (*e.g.* admissible load / energy consumed) are higher.
- **Number of collisions per segment per time step.** As mentioned previously (section 3.2.4.1), despite the use of passive avoidance, the single point per segment

constraint is not sufficient to ensure a strict collision avoidance. As a result, and despite the initial specifications, the number of collisions cannot be ignored in the indicators. However, the use of the CCC has a significant impact on the consequences of the introduction of this indicator in the problem as discussed in section 4.2.4.3.

There are no purely conflicting indicators in the proposed set. The minimization of the number of DOFs and the robot total length tends to be in harmony (robot size minimizers). They together tend to be in conflict with the minimization of the maximum linear error along the trajectory tracking. The number of collisions is a bit particular: it is conflicting with the robot size minimizers for small robots (no redundancy) and for big robots (cluttered environment). However, the CCC tends to minimize its impact as only a few robot encounter collisions, due to singularity in the avoidance method, which can be interpreted as an independency feature for this criterion.

4.2.3 Control law

The control law used in the simulation of the fitness function is an essential element. It is responsible of a part of the scores (trajectory tracking errors and collisions) the individuals get in the evaluation.

4.2.3.1 Control law expected features

The control law gives a behavior to the individuals. Many features are demanded:

- **Genericity in robots**. Each morphology the genome can describe must be controllable by the control law without any specific restriction (geometry, degree of freedom, ...). The skills of each morphology must be exploited with equal chances;
- **Genericity in situations**. The control law must take into account and manage appropriately the specific conditions as configurations singularities, constrained areas, oscillating behaviors, ...;
- **Representativity**. The control law used in the fitness should produce realistic behaviors to get meaningful evaluations and meaningful results. In particular, collisions should not occur in the simulations, as it cannot occur in reality;
- **Coherency with indicators**. Robots are fairly evaluated when the control law takes the fitness criteria into account to act appropriately. For example, if the number of collision is a part of the evaluation, the control law must integrate a collision avoidance technique. As a consequence, the control law should be able to consider several tasks.
- **Efficiency**. As a huge number of individuals are evaluated, the control law must not be time-consuming.

4.2.3.2 Context and choices

The CCC is the control law retained for the evaluations. After some design process attempts, it is decided to consider only the following constraints:

- **Joint velocity limits**. These constraints are retained as they enforce the assumption of small displacement required by the use of linearized models (Jacobians) - Control constraints Eqs. (2.23) and (2.24);
- **Obstacles**. These constraints are needed to have a realistic behavior (no collisions and active avoidance when possible) - Control constraints Eq. (2.25).

The joint position limits are not retained as they increase the size of the problem by imposing extra parameters for each joint (minimum and maximum boundaries)[5]. As a result, no joint position limit is imposed to any robot; the joint boundaries are set up adequately on the eventually retained robots.

The CCC law retained is Eq. (4.3), as in section 4.1.1 (reminded hereafter)

$$\begin{aligned}\dot{\boldsymbol{q}} =& P_{J_c}(J_O P_{J_c})^{+,DLS}\dot{\boldsymbol{X}}_O \\ &+P_{\left[\begin{smallmatrix}J_c\\ \bar{J}\end{smallmatrix}\right]}(J_{\bar{c}}\ P_{\left[\begin{smallmatrix}J_c\\ \bar{J}\end{smallmatrix}\right]})^{+,DLS}(\dot{\boldsymbol{X}}_{\bar{c}} - J_{\bar{c}}P_{J_c}(J_O P_{J_c})^{+,DLS}\dot{\boldsymbol{X}}_O)\end{aligned} \tag{4.8}$$

- **Operational input**: the operational input $\dot{\boldsymbol{X}}_O$ is computed by the difference between the current effector position and the current desired effector position divided by the time step period δt (which value is transparent as the simulation is at the velocity kinematic level).
- **Pseudo-inversion**: the Damped Least Square pseudo-inversions are done with the damping factor λ chosen as 0.50 (see appendix B.2.2 and section 4.1.1.2).
- **Active avoidance**: the active avoidance threshold is fixed to $d_{ActAvCCC} = 40$ mm, the avoidance coefficients gains are proportional (factor $\mu = 2.5\ \ 10^{-3}$) to the inverse of the distance to the constraint; the maximum value of the avoidance magnitude $\dot{\boldsymbol{X}}_C$ is fixed to $\rho_{max} = 0.25$ which is equivalent to a distance of $\delta = 10$ mm between the manipulator closest point and the obstacle ($\rho_{max} = \frac{\delta}{d_{ActAvCCC}}$).
- **Distance computation**: as mentioned in section 3.2.4.1, given the unreliability of the avoidance techniques based on a unique point per segment constraint in discrete time, an envelope of 20 mm is added around the environment.
- **Joint velocity limit**: $\dot{\boldsymbol{q}}_{max}$ has been fixed to 2 rad/s.

The resulting control law is adaptable to all the potential robots as its writing does not involve gains explicitly dependent of the robots morphology (*genericity in robots*). There is no identified situation where it penalizes arbitrarily the robots (*genericity in situations*). The joint trajectory obtained for each robot is geometrically acceptable and its tracking could be considered with any velocity profile, so the simulation is meaningful (*representativity*). The CCC implemented is multi-objective: it tracks the trajectory and avoids obstacles actively, so the GA indicators are taken into account in the control law (*coherency with indicators*). Finally, the statistics obtained along the design processes shows that the mean fitness time is 300 ms. As the trajectory counts 360 point, and the presimulation counts 40 points, it gives a mean time step computation time of 0.75 ms on a off-the-shelf computer. This is considered acceptable (*efficiency*) as, even if the robots can have between 1 and 16 DOFs, the mean time step is compatible with real time requirements.

[5] As a additional justification, it is observed that almost every optimal individual obtained uses its joint on a reasonnable (*i.e.* technologically acceptable) range

4.2.4 Results and Analysis

4.2.4.1 Preliminary results

Preliminary designs have been carried out with a simple trajectory to set up the process properly. The robot presented in Fig. 4.18 is one of the optimal solutions obtained with only 3 indicators (maximum linear error: 8 cm, number of DOFs: 5, collisions: 0). As the size of the robot is not taken into account in the minimization, the robot obtained is huge (total length: 6.40 m).

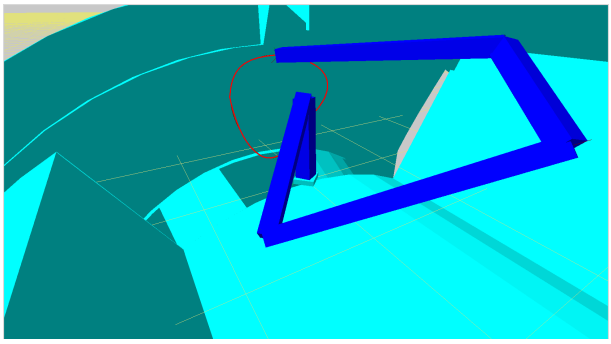

Figure 4.18: Robot 1, evaluation with linear error - number of DOFs - number of collisions.

In order to obtain more reasonable robots, the indicator of robot total length is added to the set of indicators. As a result, the robot presented in Fig. 4.19 is much shorter (total length: 1.60 m), has the same number of DOFs, it has approximately the same linear error (maximum linear error 9 cm) and never collides either.

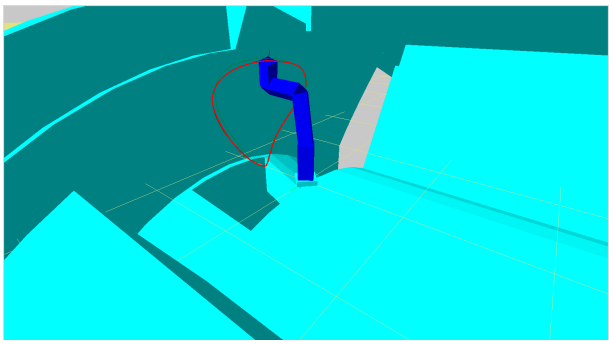

Figure 4.19: Robot 2, evaluation with indicators: linear error - number of DOFs - number of collisions - robot total length.

4.2.4.2 Final results

As the solutions fit the specifications for the simple trajectory presented above, a similar work is carried out with the trajectory representing a maintenance mission (inspection of the cutter head). Using the 4 previous indicators (linear error, number of DOFs, total length and number of collisions) seems sufficient to obtain appropriate robots. One of those is presented in the sequence of Fig. 4.20. This robot has 5 DOFs, it has a maximal linear error of 12 cm, its length is 2.80 m and it never collides.

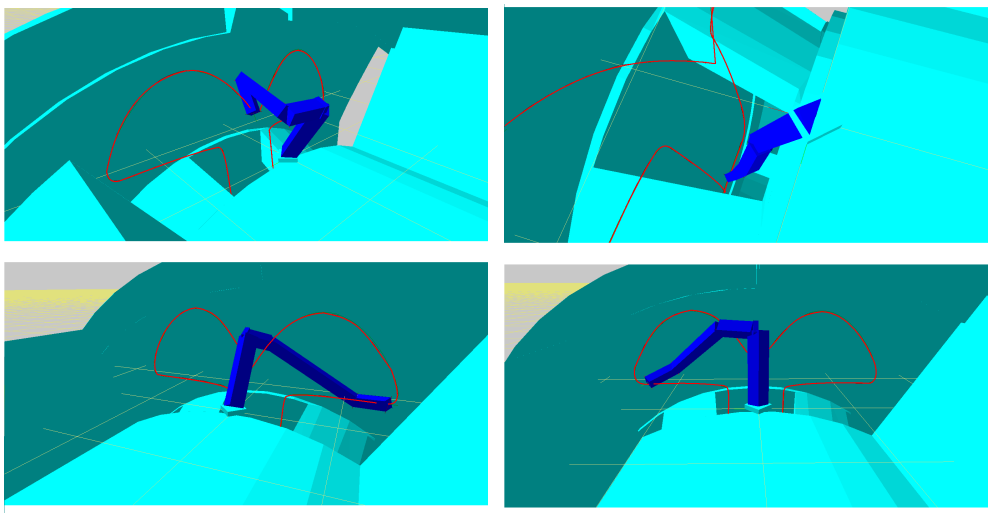

Figure 4.20: Sequence of the complex trajectory tracking. Evaluation with linear error, number of DOFs and robot total length.

An update in the TELEMACH project introduces a mobile magnetic platform that enables a vertical motion of the robot basis. This module being integrated in the design process, new results are obtained. The final retained robot has 4 degrees of freedom, it has a maximum tracking error of 10 cm, a length of 2.05 m and it never collides (Fig. 4.21).

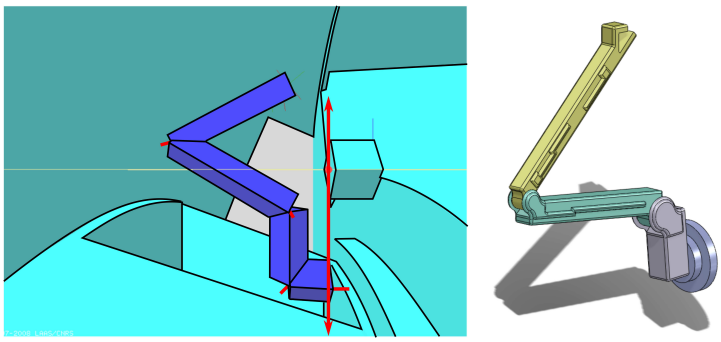

Figure 4.21: Robot obtained with a (not represented) vertical prismatic link at the beginning of the chain. This prismatic link enables the robot basis to move vertically from the red point. Given the desired trajectory, it reduces the DOFs and the lengths with respect to the robot obtained previously (Fig. 4.20).

4.2.4.3 Discussion

One of the main problems encountered when using genetic algorithms is the size and the shape of the Pareto front to be exploited, which are critical for the algorithm convergence toward appropriate solutions. Moreover, despite the use of the CCC, the number of collisions had to be introduced as a fourth indicator to be minimized, which could tend to dramatically increase the complexity of the problem. However, the CCC considerably reduces the number of collisions with respect to usual control laws. As a result, almost

every robot obtain 0 (no collision), and the size of the problem is not that much increased by the presence of this indicator. The collision occurrence may seem arbitrary for a robot morphology, as it can be seen basically as a control singularity. However, an analysis of these singularities shows that they only appear in case of insisting motions toward obstacles. As a result, it occurs for robots having generally bad scores. In order not to penalize the individuals for which this failure occurs, it has been decided not to take this indicator as a constraint (in the GA meaning: criterion which, if not respected, disqualifies the individual).

Finally, the control law used in the fitness has an impact on the size and the shape of the Pareto front from 2 sides:

- criteria nature: by reducing the collision occurrence to "accidental", this indicator is at its optimal value (no collision) for a high majority of individuals, which decreases the pareto front size;
- criteria relations: by reducing the collision occurrence to a control singularity, it results that this indicator is independent from the other ones.

4.3 Constraints compatibility experiments

This section is dedicated to experiments involving a 6DOFs manipulator submitted to various sets of constraints. The aim is to illustrate the relevance of the compatibility framework exposed in chapter 2. The results are composed of 3 experiments showing:

- the safe behavior obtained thanks to the resolution of the joint constraints compatibility;
- the safe behavior obtained thanks to the resolution of the joint constraints compatibility and the maximum joint deceleration Alternative Safe Behavior;
- the safe behavior obtained thanks to the resolution of the joint constraints compatibility and the mixable joint deceleration ASB with the Smooth Avoidance Technique.

All the experiments are done with the same state-of-the-art control law (Quadratic Programming). At each time step, the operational input sent is a desired velocity issued from a 3-DOF desired operational point (position only, no orientation). It induces a Degree Of Redundancy (DOR) of 3.

4.3.1 Experiments presentation

4.3.1.1 Experiment site

The experiments were performed in a facility of the French Atomic and Alternative Energy Commission (CEA), a government-funded technological research organization. Its Direction of Research and Technology division main goals are to meet the needs of industry by setting up industrial partnerships, promoting technology transfer and creating start-ups, and to explore and propose new innovative breakthrough technologies by putting the skills of its research laboratories to the best possible use. The 6-DOFs arm used in these experiments is a 100daN advanced remote hydraulic manipulator

dedicated to heavy loads manipulation (100 daN) with force feedback capabilities, the Maestro[6] (David *et al.* 2007), designed by CEA. Its design prevent any self collision. This robot is usually used in various applications where remote handling with high strength and dexterity are needed like in nuclear or offshore hostile environments.

4.3.1.2 Experimental equipment

The robot's controller uses a generic hard real-time application, TAO2000 (Gicquel *et al.* 2001), developed by CEA for computer aided teleoperation systems (teleoperators) and coming from its experience for objects remote manipulation in hazardous environment. It can address both masters and slaves robots, whatever their kinematics and actuation technologies, providing them with a whole generic set of useful features with nearly no specific development. This application provides, via a standard Ethernet link, a high level communication interface to control the robot and a low level real-time tuning and spying interface.

The Maestro works at the front of a tunnel boring machine mock-up (Fig. 4.22).

Figure 4.22: Teleoperated Maestro operating in front of the tunnel boring machine cutting wheel mockup.

4.3.1.3 Initial assumptions versus experimental conditions

The assumptions made in chapter 2 are at the basis of the framework (section 2.1.4). The validity of these assertions and their actual potential violation should be discussed. They are briefly recalled hereafter:

1. The perception of the system is exact;
2. The models of the system and the environment are known exactly;
3. The system limits implemented in the controller are representative of the system real capabilities;
4. Once the control problem expressed is *feasible* (all the constraints can be satisfied simultaneously), an algorithm is able to solve it without any constraint violation;

[6] http://www.cybernetix.fr/Hydraulic-arms

5. At a given time step, a desired control input (*e.g.* $\boldsymbol{u}(k) = \dot{\boldsymbol{q}}_{des}(k+1)$ for time step k) satisfying the constraints is perfectly carried out at the next time step ($\dot{\boldsymbol{q}}_{des}(k+1) = \dot{\boldsymbol{q}}(k+1)$).

Assumptions 1, 2, 3 and 5 generate errors: sensors have a limited precision and their measures are influenced by the environment variations; there are always geometric uncertainties and many factors (*e.g.* oil temperature for an hydraulic robot) may impact the quality of the model; even if the actuators capabilities are generally known, a whole robot controller usually interlinks many loops, which may complicate the problem, especially through the response time for example. Thus, these assumptions imply errors and approximations. Despite the work carried out on safety, these approximations may provoke incompatibilities. These incompatibilities are localized and do not have a big impact on the robot behavior: as shown on the following results, the envelope needed to absorb them could be small with respect to what would be needed without the compatibility study. However, at the control level, an incompatibility provokes the impossibility to solve the problem. For practical reasons, the occurring incompatibilities are *denied* at the control level: for example, if the current position of the joint parameter $q_3(k)$ is inferior to the artificial minimum joint position $q^{art}_{3,min}$, then the inferior joint position limit is taken as the minimum between the current joint position and theoretical minimum joint position $q_{3,min}$ gets $\min(q_3(k), q^{art}_{3,min})$.

Assumption 4 is true: many control resolution problem algorithms can be used. As an example, a generic QP solver has been used.

Finally, the control constraints validity is assumed, which involves that the errors induced by finite differences approximations used in transpositions across physical levels are neglected. Independently of their magnitude these errors nonetheless provoke incompatibilities. However, in the same manner than for the previous assumptions, these incompatibilities are absorbed by envelopes and denied for the solver. In the particular case of ASB, these errors are cumulated over all the prediction toward a stop, which typically represent 100 time steps (deceleration 1 rad/s^2, velocity 1 rad/s, period 10 ms). In this case, 2 kinds of errors are added: 1/ the errors induced by the approximation of finite difference, which should not exceed the distance traveled during 1 time step at maximum speed (see Fig. 4.23); 2/ the absence of matching with reality during the prediction, which may cause more important errors due to the approximations induced by assumption 3.

From a practical point of view, the envelopes around the joint position limits e_j and around the environment e_c are unknown from the controller and considered as an origin offset: for example, the controller considers that a collision occurs if the distance between the robot and the environment is lower than e_c.

4.3.2 Safe behavior with compatible constraints - experiment 1

This first experiment[7] illustrates the behavior of a multi-body robot submitted to control constraints modified to become compatible.

4.3.2.1 Task presentation

The robot is subject to a brutal fold up from a configuration of extended robot to a configuration in which the robot has reached its joint position limits (Fig. 4.24). During

[7] http://www.isir.upmc.fr/UserFiles/File/VpadoiS/Medias/JointPosLim.avi

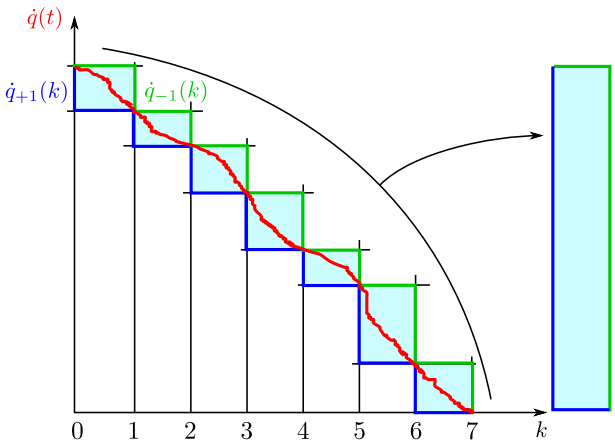

Figure 4.23: Approximations induced by finite differences. $\dot{q}(t)$ is a continuous velocity, $\dot{q}_{+1}(k)$ its approximation with advance convention $q(k) = \dot{q}(k)+q(k-1)$, whereas $\dot{q}_{-1}(k)$ is its approximation with delay convention $q(k) = \dot{q}(k-1) + q(k-1)$. In any case, the maximum error on position along this deceleration is the distance carried out during one time step at initial velocity.

a first period (5.0 s), the desired operational velocities are maintained constant and maximum toward a point at the left infinite; then, the desired operational velocity brings back the robot toward the initial Cartesian point at lower velocity (the aim is to check that the deceleration toward the joint position limit is safe). The considered e-state constraints are joint position, velocity and acceleration limits (Eqs. (2.13) - (2.18)). For the sake of clarity, the e-state constraints limits are the same for each joint: respectively $\pm 1.0\ rad$, $\pm 1.5\ rad/s$, $\pm 2.0\ rad/s^2$. The acceleration limits have been taken voluntarily low (lower than the robot actual capabilities) in order to better illustrate the results. For this particular experiment, the trajectory is considered in a (z,y)-plane (2 DOF desired velocity) and only 3 DOFs are used (see Fig. 4.24), which brings the DOR to 1.

The control constraints used to enforce the considered e-state constraints are F_V, F_A and $F_{P'}$ (respectively Eqs. (2.23), (2.24), (2.26), (2.27), (2.29) and (2.30)). The control problem is expressed as a QP, and the solver is an efficient open source algorithm[8].

Given the limits of the robot, the joint position overshoot could reach 0.56 rad without the proposed methodology (by taking Eqs. (2.21) and (2.22) as the joint position control constraints for example). As a benefit of our approach, the envelope retained on joint position limits for this experiment is $e_j = 0.1\ rad$.

4.3.2.2 Results and analysis

The results are presented on Fig. 4.25. The $\dot{\boldsymbol{q}}_{des}$ values are the joint velocity sent to the actuators ($\dot{\boldsymbol{q}}(k+1)$) and $\dot{\boldsymbol{q}}_{real}$ are the velocities actually carried out by the actuators. Only the 3 DOFs concerned by the planar trajectory are represented (the other are excluded from the model, so they remain static). During the first second, each joint contributes to the operational motion at its best: accelerations are maximal for each joint. Joint 5 is the first to undergo a deceleration (before reaching its maximum velocity) due to the initial proximity to its position limit. Joint 3 reaches its velocity

[8]QuadProg++: http://sourceforge.net/projects/quadprog/

Figure 4.24: Views of the robot in initial position (extended) and at $t = 5s$ (fold up). The white arrow shows the direction of the constant operational desired velocity from $t = 0\ s$ to $t = 5\ s$.

limit for a short time. Joint 2 does not perform high accelerations due to the fact that the operational velocity is sufficiently high thanks to the other joints. At $t = 4.0\ s$, the small motion of joint 2 is induced by a disturbance which, given the system configuration, leads to the tracking of the desired Cartesian velocity. At $t = 5.0\ s$, the operational desired velocity is inverted (the robot goes back to its initial operational position), and the robot gets away from its boundaries without any difficulty. At the end of the experiment, ($t = 7.5\ s$), the deceleration is provoked by a reduction of the operational desired velocity; it is not provoked by any constraint. The envelope violation of joint 3 occuring at the beginning of the experiment ($t = 1.5\ s$) is attributed to the approximations discussed in section 4.3.1.3 (especially the exact execution of the desired joint input). However, the envelope is slightly violated, which tends to show that it can reliably be reduced (maximum overshoot is $3.03E^{-2}\ rad$).

4.3.3 Safe behavior with ASB - Experiment 2

The second experiment[9] illustrates the behavior of a multi-body robot submitted to incompatible control constraints; at each time step, the computed control input is sent if a consecutive deceleration toward a stop e-state is admissible (see section 2.4.5).

4.3.3.1 Task presentation

The robot is subject to various motions in the cluttered environment of the cutting wheel mock-up (Fig. 4.22). The desired operational velocity is issued from a 3D trajectory involving unreachable points. The considered e-state constraints are joint position, velocity and acceleration limits and collisions avoidance (Eqs. (2.13) - (2.18) and (2.20)). The trajectory involves motions close to joint position limits. The joint position limits are $\pm 1.0\ rad$. The joint velocity limits are not reached during this experiment; The joint accelerations limits are set to $1.0\ rad/s^2$. The distances are computed in real-time using a CAD model of the environment (Fig. 4.22).

The control law is similar to the previous experiment. To deal with incompatible constraints (joint acceleration limits and collisions avoidance), the control uses algorithm

[9] http://www.isir.upmc.fr/UserFiles/File/VpadoiS/Medias/ObstAvoidASB1.avi

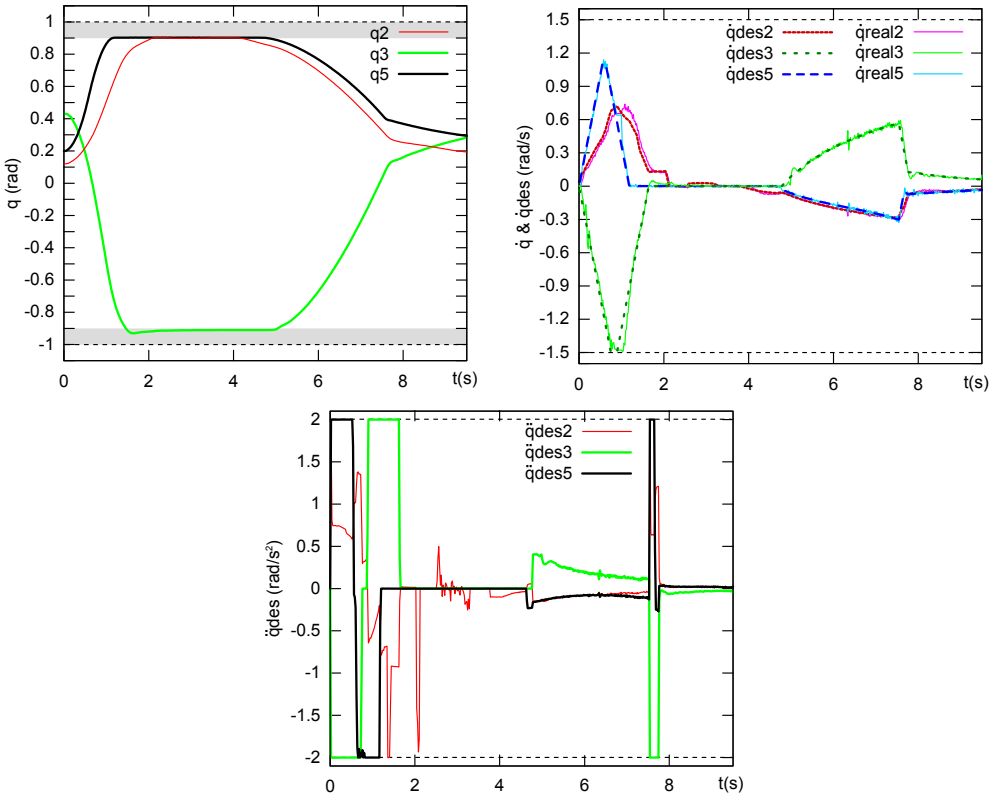

Figure 4.25: Position, velocity and acceleration of joints 2, 3, and 5 during experiment 1. The position is directly measured on the robot, the velocity $\dot{\boldsymbol{q}}_{des}$ is the input sent to actuators and $\dot{\boldsymbol{q}}_{real}$ is the measured one. The acceleration is computed from $\dot{\boldsymbol{q}}_{des}$. All the variables remain between their limits. The control constraints modification imposes appropriate decelerations to satisfy the joint position limits.

1. To differentiate accelerations due to the trajectory tracking and accelerations issued from ASB1, the acceleration value for prediction and alternative behavior is lower than the one retained for the control constraint: 0.9 rad/s^2. This modification has no major impact on the results but makes them clearer.

Given the limits of the robot, the joint position overshoot could reach 0.5 rad without the proposed methodology. In the same conditions, given the dimensions of the robot, the potential collision without ASB would have required an envelope $\sim 1\ m$ to be avoided. As a benefit of our approach, the envelope retained on joint position limits is $e_j = 0.1\ rad$ and the envelope around the environment is $e_c = 0.1\ m$.

4.3.3.2 Results and analysis

The results are presented on Fig. 4.26. The 4 motions getting close to obstacles, easily identifiable at $t = 2.0\ s$, $t = 5.0\ s$, $t = 8.0\ s$ and $t = 17.0\ s$ on the graph of distance to environment, end-up in the envelope e_c. The 3 first motions ($t = 2.0\ s$, $t = 5.0\ s$ and $t = 8.0\ s$) gets close to obstacles with a reasonable velocity, but as there is no compatibility between collisions avoidance and acceleration limits, resort to the alternative behavior is needed (red squares). The fourth motion toward obstacles is done at higher speed; the deceleration begins nearly 1.0 s before the impact (blue square). Finally a motion in the neighborhood of the obstacle generates high frequency oscillations on the acceleration ($t = 20.0\ s$). Actually, as the robot remains close to the obstacles, a deceleration at the joint level tends to maintain the robot close to the environment. As a result, oscillations between the trajectory tracking and the alternative behavior occur. Thanks to the envelope e_c taken, safety is preserved.

4.3.4 Integration of a SAT into the mixable joint deceleration ASB - Experiment 3

This third experiment[10] illustrates the possibility to introduce usual collisions avoidance methods into a safe framework for a multi-body robot. The resulting behavior remains safe and takes full advantage of the avoidance method.

4.3.4.1 Task presentation

The robot is in charge of reaching a setpoint from which it is separated by an infinite horizontal plane (Fig. 4.27). As in the first experiment, the trajectory is considered in a plane (2 DOF desired velocity) and only 3 DOFs are used (the same as in Fig. 4.24), which gets the DOR to 1. The magnitude of the desired velocity is maintained constant toward the desired point. At the end of the experiment, a second setpoint is given to get far from the obstacle. The considered e-state constraints are joint position, velocity and acceleration limits and collisions avoidance (Eqs. (2.13) - (2.18) and (2.20)). The trajectory does not involve motion close to joint position limits. The joint velocity limits are not reached along this motion. The joint accelerations limits are set to 1.0 rad/s^2. The distances are computed in real time using a virtual environment.

The control law is based on the one of experiment 2. The approach used to preserve safety in the previous experiment has the severe drawback to generate oscillations on the accelerations when the robot is moving along obstacles. Actually, the control alternates

[10] http://www.isir.upmc.fr/UserFiles/File/VpadoiS/Medias/ObstAvoidASB2SAT.avi

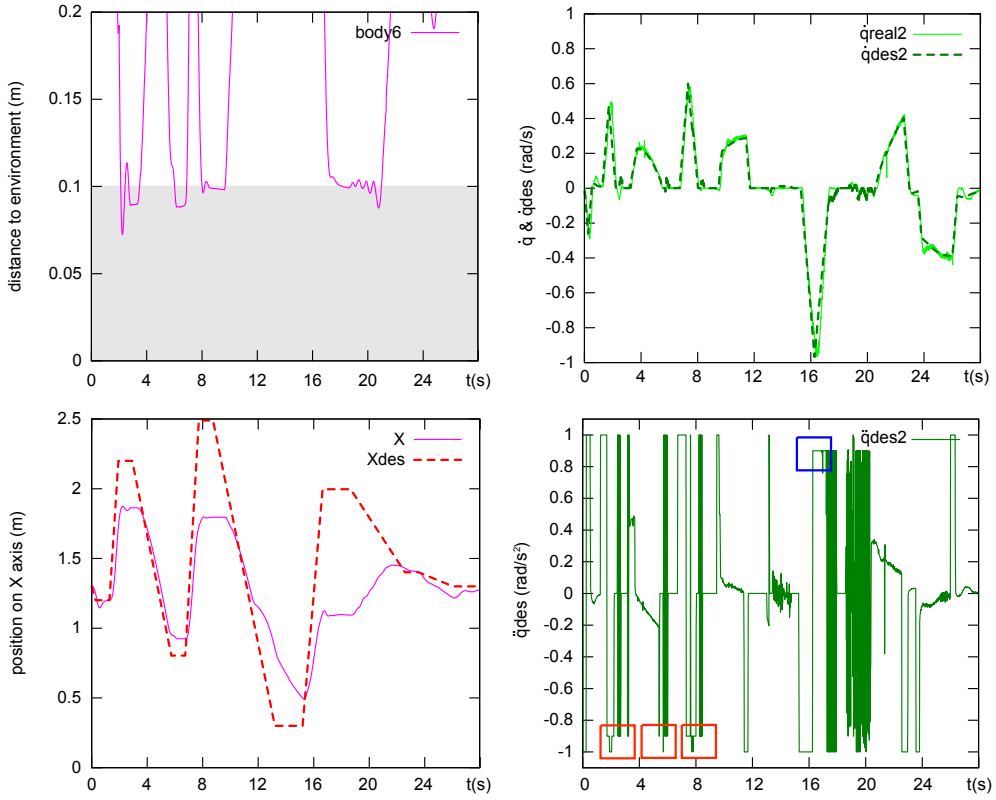

Figure 4.26: Results of experiment 2. *Left column*: shortest distance between body 6 and the environment, desired and real position along cartesian axis X; *right column*: desired and real velocity of joint 2, accelerations of joint 2.

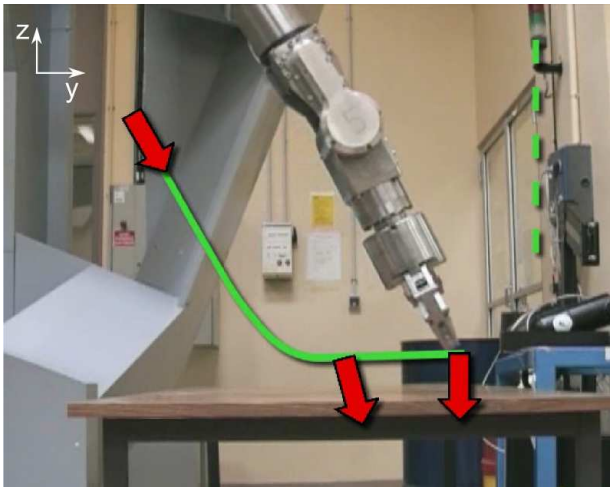

Figure 4.27: Views of the robot during the trajectory. The red arrows show the desired operational velocity input along the robot trajectory (green line).

between the trajectory tracking and the alternative behavior at nearly each time step. As detailed in section 2.4.5, the control constraint induced by the SAT (Eq. (2.38)) is added to the set of considered constraints. The following values have been used: $d_i = 0.15\ m$ (area I), $d_s = 0.07\ m$ (area II) and the envelope around obstacles is $e_c = 0.05\ m$ (area III). Taking different values for d_s and e_c eases the interpretation of the results.

4.3.4.2 Results and analysis

The results are presented on Fig. 4.28. As in experiment 2, the arrival on the obstacle causes the maximum overshoot in the area II. The robot never enters the security envelope (area III) as it is managed by the SAT. The distance to obstacle stabilizes during the sliding motion (see Fig. 4.27) until $t = 6.0\ s$ when another objective is given to the effector. The transition time can be detected on the acceleration (blue square), when it switches from 1.0 rad/s^2 (deceleration coming from the alternative behavior) to approximately 0.93 rad/s^2. At that time ($t = 1.62\ s$), the distance to the obstacle is 10.9 cm, and the avoidance method begins to limit the robot motion along direction z. The acceleration is then smooth, the collision management being ensured by the SAT. During the motion along the obstacle (between $t = 2.0\ s$ and $t = 6.0\ s$), the velocity of joint 2 contributes to the motion, but the velocity is small as the setpoint is far under the table, increasing the angle between the desired velocity vector and the infinite plane toward orthogonality.

4.3.5 Partial Conclusion

The presented experiments illustrate the behavior of a robot which safety is ensured thanks to a modification of the constraints expression to ensure their compatibility and alternative safe behaviors in case of unsolvable incompatibilities.

In the first experiment, the manipulator is submitted to compatible constraints; the control constraints impose appropriate deceleration to maintain the joints positions within their limits. In the second experiment, the manipulator is submitted to a set of constraints which are known to be incompatible. Various motions close to obstacles

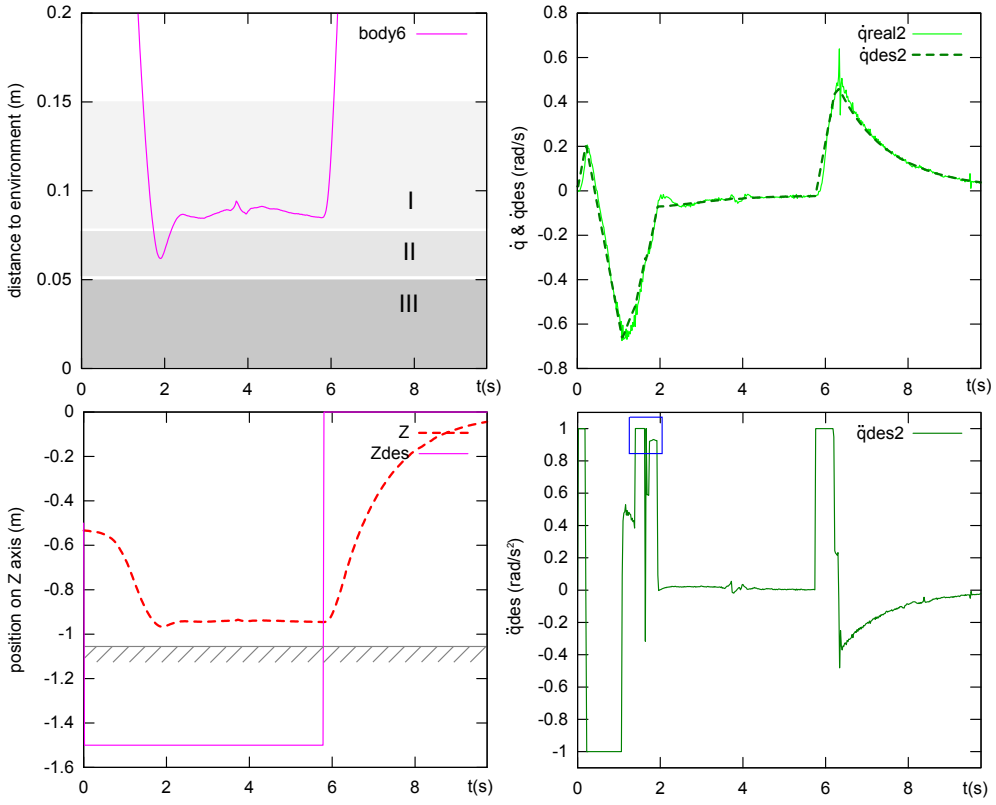

Figure 4.28: Results of experiment 3. *Left column*: shortest distance between body 6 and the environment (areas I, II and III are limited by parameters d_i, d_s and d_e), desired and real position along Cartesian axis Z; *right column*: desired and real velocity of joint 2, accelerations of joint 2.

provoke critical situations where the maximal deceleration Alternative Safe Behavior is exploited. The obtained behavior is safe but oscillations may occur because of the ASB, which is not completely satisfying. The third experiment focuses on an obstacle avoidance when incompatibilities are treated with mixable deceleration ASB. The obtained behavior is no longer oscillating.

As a result, whatever the set of constraints, the behavior is safe and present satisfying properties (smoothness, no oscillation, no ill-timed use of ASB). It is important to keep the resort to ASB as rare as possible, because it decreases the quality of solutions in terms of objectives tracking; ASB should remain a kind of emergency solution in control.

Chapter 5

Conclusion

5.1 Contributions

In this work, we tackle the formulation and the resolution of the control problem of constrained robots. Our contributions to this fairly complex and large issue are methodological but also applied, in a rather generic manner, to dedicated robotic applications: constraint-based robot design and teleoperation.

In the first chapter, the context of this thesis is presented from the industrial point of view of the TELEMACH project. This project dedicated to the automation of maintenance tasks in the cutter-head of tunnel boring machines, exhibits general issues in terms of constraint-based robot design and teleoperation. The link between these two fields of investigation is drawn: they both rely on the safe and optimal reactive control of robotic manipulators. A short overview of the literature in this domain is proposed and our contribution to open problems is presented. This contribution goes beyond what is usually proposed in the state-of-the-art: not only do we tackle the control problem resolution but we also address its formulation. The latter is very critical since it conditions the existence of a solution to the control problem but also has some interesting implications in terms of its resolution.

In the second chapter, the formulation problem is introduced from a safety preservation perspective. Indeed, ensuring the existence of a solution to the control problem that complies with robotic constraints is a safety issue both for robots and their environments. We showed that to ensure safety either the constraints expressions have to be permanently compatible, *i.e.* allow some space for control at any time, or alternative safe behaviors (ASBs) must be designed in order to properly manage emergency cases where the robot has to be stopped in a safe manner. As examples, the study of all the combinations of the set {joint position limits - joint velocity limits - joint acceleration limits - obstacles avoidance} is proposed. As a result, the intuitive expression of joint position limits is modified to remain compatible with joint accelerations limits, and ASBs are proposed to ensure safety when dealing with obstacles avoidance[1]. A particular strategy is developed to take full advantage of the usual avoidance techniques while maintaining safety.

The control problem being adequately formulated, the third chapter focuses on its resolution. Our proposed resolution algorithms have various specifications: 1/enforce

[1]Compatibility for these constraints is hard to prove, due to the dependence of operational deceleration capabilities with respect to the robot configuration

compliance with the considered constraints; 2/fit the context efficiency requirements (real time, simulation, etc.); 3/find the optimal solutions; 4/offer a satisfying general behavior. The compliance with constraints being a prerequisite, compromises between efficiency (time computation), optimality (reduction of the tasks errors) and correct behavior (no oscillations, smoothness) are proposed. First, the elaboration of the Constraints Compliant Control law (CCC) based on the passive avoidance principle and existing techniques coming from analytic model inversion and convex optimization methods enables to obtain robust, efficient and interesting characteristics. The passive avoidance principle and the solutions scaling enables to overcome the drawbacks of active avoidance at the top (optimality loss, oscillations) or at the bottom (constraints violation, infinite terms) of the hierarchy. The displaced configuration method extends the Constraint Compliant Control law to any set of constraints and proposes a compromise between optimality and efficiency.

The fourth chapter is dedicated to applications of the proposed framework. First, simulation results illustrate the CCC performances in various contexts. When involving joint position limits, velocity limits and obstacles avoidance, the control law has a behavior similar to state-of-the-art control laws in underconstrained cases. In over-constrained cases, it fulfills the multiple objectives while satisfying the constraints without the drawbacks encountered with state-of-the-art control law (high lack of optimality and oscillations). If the constraints set prevents the admissibility of the current configuration, the use of a virtual displaced configuration in the control problem resolution enables to obtain a compromise between efficiency and optimality (safety being ensured anyway) through a safe single iteration resolution method. It is particularly adapted to the task based design of a manipulator morphology in cluttered environments, for which a huge number of robots are evaluated through trajectory trackings, for which only kinematics constraints hold. The impact of the CCC on the task based design is discussed and oriented toward efficiency. The obtained robot morphologies are discussed with respect to the associated context of tunnel boring machine maintenance. The methods relative to safety developed in this thesis have been applied on a 6-DOF manipulator operating in a cluttered environment. The obtained results illustrate the reliability of the approach and validate the expected performances: whatever the set of constraints, the behavior is safe and present satisfying properties (smoothness, no oscillation, limited use of ASBs).

5.2 Perspectives

The rather large scope of the research topic tackled in this thesis leaves many unanswered questions and hence several directions for future research. We briefly discuss these directions here.

5.2.1 Safe control problem formulation

- **Transposition of the work on constraints compatibility to torque controlled systems.** The control of robotic systems involving fast motions and thus implying high dynamics requires resorting to the dynamic equations of motion of the system and displaces the control problem at the torque level. The transposition of the proposed work at this physical level is rather straightforward since there exists a bijective mapping between torques and joint space accelerations. In fact, actuator limitations are more naturally expressed in terms of torques limits

since acceleration limits are configuration dependant. As a matter of fact, this perspective is a short term one. Its impact may be significant, as more and more robots are being controlled at the torque level.

- **Adaptation of the work on Inevitable Collision State to Inevitable Violation State.** The notion of Inevitable Collision State introduced by Fraichard (2004) denotes a state for which, whatever the control inputs sent, a collision finally occurs. It is extended in this thesis to Inevitable Violation State, denoting a state for which, whatever the control inputs sent, a constraint violation finally occurs. The ICS notion is now widely exploited in robotics control and it would be profitable to apply some contributions in this domain to the IVS studies, for example:
  - Management of dynamic environments. Dealing with moving obstacles is particularly complex in our case: as the constraints compatibility depends on the prediction of the system motion capabilities with respect to the environment, it strongly depends on the assumptions over the environment dynamics. As a consequence, a work on the specifications and on the possible assumptions of the problem may be carried out. A first perspective in that field is to deal with self-collisions;
  - ICS-checkers and avoiders (Parthasarathi *et al.* 2007, Martinez-Gomez *et al.* 2008, 2009);
  - Approximations of ICS regions (Chan *et al.* 2008);
  - Probabilistic checking (Bautin *et al.* 2010).
- **Intelligent Alternative Safe Behaviors.** Although efficient, the ASBs proposed in the thesis are rather basic: the maximum deceleration ASB is intuitive and simple but may lead to oscillations; the mixable deceleration ASB offers better performances. Actually, ASBs are roughly considered in this thesis as emergency stops at the control level. However, they can be considered as IVS avoiding motions, or transitions between determined safe regions. In that scope, they can even take the tasks objectives into account, thus getting closer to the planning domain. In that framework, the emergence of real-time motion planning (Brock *et al.* 2000) and Partial Motion Planning (Petti *et al.* 2005) which focuses on the optimal use of the time step as a computation time are promising.

5.2.2 Control problem resolution

- **Constraints criticity.** As mentioned in the conclusion of section 3.2, a strong perspective of the CCC is to rely on the KKT optimality conditions to increase efficiency and optimality. This would enable to find the optimal combination of constraints (Optimal Active Constraints Set - OACS) efficiently, relying on the widely known active sets method of the convex optimization research field. However, there are many cases where the number of constraints is huge and even optimized algorithms may turn unable to find optimal solutions within the time step in real time. In Escande *et al.* (2010) for example, the robot HRP-2 is controlled in real time (40 DOFs, at least 120 constraints, so 2^{120} possible combinations). In the case of multiple simultaneous constraints activations, many iterations may be

required. The key is to guess the OACS at the first iteration of the control problem resolution. This research for the optimal active constraints set could be fed by the knowledge of the system evolution within its environment: as an example, when a joint is close to its maximum position limit, considering the constraints of minimum joint limit as potentially active has no sense. As a consequence, a lot of constraints can be intuitively not considered. A first idea is to take as a first guess the OACS of the previous time step. But in addition, there are many reasonable assumptions to make: one can think of a constraints criticity index, quantifying the probability to have the constraints in the OACS or not.

- **Integration of the environment shape in the dangerousness estimation for collisions avoidance**. The usual collisions avoidance method, as proposed by Maciejewski *et al.* (1985), has an avoidance coefficient magnitude linked to the distance to the environment, as it is considered that the closer the robot is to the environment, the more dangerous it is. More recently, Choi *et al.* (2000) proposed to include the velocity toward obstacle into the avoidance motion magnitude (*collidability*), as it contributes to the criticity of a situation. However, these methods consider elements involved in the danger estimation, but they do not consider danger formally, as it is done for example by Ikuta *et al.* (2003) or Kulic *et al.* (2006). As an example, it seems obvious that, given the method to avoid obstacles (impose an avoidance effort or velocity to the closest point of a part of the robot), the shape of the environment and in particular its convexity are involved in the criticity of a situation. Moreover, it is closely linked to the time period of the controller with respect to the velocity of the robot. Finally, another way to tackle this problem is not to avoid obstacles directly, but rather to avoid inevitable collision states, as they include all of what can be meant by the "dangerousness" of a situation.

Appendix A

Compatibility between joint position and acceleration limits

The relations between position, velocity and acceleration in discrete calculus are different from the continuous ones. Let $(q(k))$ be the sequence of positions of the joint of a 1-DOF system along the iterations, $(\dot{q}(k))$ its velocities and $(\ddot{q}(k))$ its accelerations. The motion is modelized with a first order approximation by

$$\begin{aligned} q(k+1) &= q(k) + \dot{q}(k+1)\delta t \\ \dot{q}(k+1) &= \dot{q}(k) + \ddot{q}(k+1)\delta t \end{aligned} \tag{A.1}$$

As shown by Decré *et al.* (2009), in case of constant acceleration $\ddot{q}_m$, the position evolution in s time step is

$$q(k+s) = q(k) + s\dot{q}(k+1)\delta t + \frac{1}{2}(s^2 - s)\ddot{q}_m\delta t^2 \tag{A.2}$$

If we suppose $\dot{q}(k) > 0$ and $\ddot{q}_m < 0$ (deceleration example), the condition $q(k+s) \leq q_M$ for all integer s leads to

$$\dot{q}(k+1) \leq \frac{q_M - q(k)}{s\delta t} - \frac{1}{2}(s-1)\ddot{q}_m\delta t \tag{A.3}$$

The minimization of the right member can be obtained by relaxing the integer optimization problem ($s \to s_{\mathbb{R}}$) and differentiating this expression with respect to $s_{\mathbb{R}}$

$$\left.\begin{array}{rl} s_{\mathbb{R}} & \geq 0 \\ -\frac{q_M - q(k)}{s_{\mathbb{R}}^2\delta t} - \frac{1}{2}\ddot{q}_m\delta t & = 0 \end{array}\right\} \Leftrightarrow s_{\mathbb{R}} = -\frac{\sqrt{-2\ddot{q}_m(q_M - q(k))}}{\ddot{q}_m\delta t} \tag{A.4}$$

which then can be solved by finding the integer value $s_{\mathbb{N}}$ that minimizes the maximum velocity in (A.3). However, if $s_{\mathbb{R}} < 1$, the method retained by Decré *et al.* (2009) ($\dot{q}(k) \leq 0$) may violate the acceleration limit constraint. Moreover, this method is tight and may fail in case of any measure error.

The expression of the deceleration distance obtained from the relaxed expression is

$$d_{\mathbb{R},dec}(k) = s_{\mathbb{R},dec}\dot{q}(k+1)\delta t + \frac{1}{2}(s_{\mathbb{R},dec}^2 - s_{\mathbb{R},dec})\ddot{q}_m\delta t^2 \tag{A.5}$$

with

$$s_{\mathbb{R},dec} = -\frac{\sqrt{-2\ddot{q}_m(q_M - q(k))}}{\ddot{q}_m\delta t} \tag{A.6}$$

It is obvious that $d_{\mathbb{R},dec}(k) \geq d_{\mathbb{N},dec}(k)$ where $d_{\mathbb{N},dec}(k)$ would be the exact discrete deceleration distance at time step k. To satisfy the constraints, a cautious condition is

$$d(k+1) \geq d_{\mathbb{R},dec}(k) \tag{A.7}$$

where $d(k+1)$ is the distance $q_M - q(k+1)$. It leads to

$$\dot{q}(k+1) \leq \frac{(q_M - q(k)) - \frac{1}{2}(s^2_{\mathbb{R},dec} - s_{\mathbb{R},dec})\ddot{q}_m \delta t^2}{(s_{\mathbb{R},dec}+1)\delta t} \tag{A.8}$$

The comparative deceleration of a joint with the two methods can be observed on Fig. A.1.

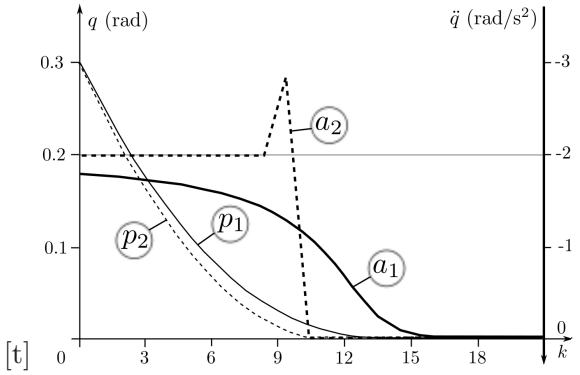

Figure A.1: Curves p_1 and p_2 represent the distances to the joint position limit ($q_m = 0$) according to the deceleration strategies respectively proposed in this work and in Decre *et al.* (2009). Curves a_1 and a_2 represent the associated deceleration profiles. The acceleration limit is $\ddot{q}_m = -2\ rad.s^{-2}$ and $\delta t = 0.05\ s$. It is obvious form this graph that strategy 1 is safe whereas 2 is not. Moreover the deceleration profile of strategy 1 is smoother than strategy 2 which requires full deceleration and thus a peak of jerk.

The choice of Eq. (A.8) as the joint position limit constraint provokes a small reduction of the reachable positions. Actually, the resolution of $\dot{q}(k+1) = 0$ in Eq. (A.8) induces

$$q_M - q(k) = \frac{-\ddot{q}_m \delta t^2}{8} \tag{A.9}$$

which means that the asymptotic value of the joint position according to this constraint is no longer q_M but

$$q_{M'} = q_M - \frac{-\ddot{q}_m \delta t^2}{8}. \tag{A.10}$$

The order of magnitude of this reduction is $\sim \delta t^2$, which can be considered negligible. However, it is a reduction of the space of reachable e-state, and all the compatibility studies involving this control constraint must be checked over the joint position space

$$\mathscr{A}_P = [\boldsymbol{q_{m'}};\ \boldsymbol{q_{M'}}] \tag{A.11}$$

where $\boldsymbol{q_{m'}} = q_m + \frac{\ddot{q}_M \delta t^2}{8}$.

Appendix B

Linear Problem Resolution Reminder

This appendix is a brief reminder of the linear problem resolution results regularly used in robotics. It is based on Ben Israel *et al.* (2003); it is also largely inspired of the report of Buss (2004) and on the thesis of Padois (2005).

B.1 Problem

A linear problem is expressed by the equation

$$\dot{\boldsymbol{X}} = J\dot{\boldsymbol{q}} \tag{B.1}$$

where $\dot{\boldsymbol{X}}$ is a vector of size m, J is a (n, m) matrix and $\dot{\boldsymbol{q}}$ is a vector of size n. The aim of the problem is to find a solution $\dot{\boldsymbol{q}}$ so that Eq. (B.1) is verified.

As a first remark, this equation does not have an exact solution if $\dot{\boldsymbol{X}} \notin Range(J)$; however, it can be interesting to find approximated solutions to this problem. Conversely, if $\dot{\boldsymbol{X}} \in Range(J)$ and $Dim(Ker(J)) > 0$, there is an infinite number of solutions (*redundancy*); it can be then profitable to exploit the possibility to choose a solution adapted to secondary needs.

B.2 pseudo-inversion

B.2.1 General expression

The problem addressed requires an inversion of the relation expressed by Eq. (B.1), even if the matrix J is not square or invertible. As exposed by Ben Israel *et al.*, the pseudoinverse is given by:

$$J^{+} = G^{T}(F^{T}JG^{T})^{-1}F^{T} \tag{B.2}$$

where $J = FG$ is the full rank decomposition of J. The solution given by

$$\dot{\boldsymbol{q}} = J^{+}\dot{\boldsymbol{X}} \tag{B.3}$$

has the following properties:

- Exact and minimal l_2-norm solution if the equation is solvable ($\dot{\boldsymbol{X}} \in Rg(J)$);

- Approximate result with minimal operational l_2-norm error if the system is not solvable ($\dot{\boldsymbol{X}} \notin Rg(J)$);

- $J^+ = J^{-1}$ if J is square and full rank.

When the matrix J is not singular, the pseudoinverse has simple expressions:

- if J is full rank in line ($rk(J) = m \leq n$), Eq. (B.2) is equivalent to

$$J^+ = J^T(JJ^T)^{-1} \tag{B.4}$$

- if J_O is full rank in column ($rk(J) = n \leq m$), Eq. (B.2) is equivalent to

$$J^+ = (J^TJ)^{-1}J^T \tag{B.5}$$

B.2.2 Approximated pseudoinverse

The problems induced by singularities are widely known and often addressed in the literature. An elegant method to avoid the drawbacks of singularities at a small precision cost is the Damped Least Square pseudo-inversion method, introduced by Wampler (1986) and Nakamura *et al.* (1986).

In the neighborhood of a singularity, (JJ^T) is badly conditioned. This conditioning problem leads to high joint space velocities[1] (potentially infinite). To avoid, the problem can be slightly changed to induce a conditioning improvement of (JJ^T). The proposed modification is

$$\min_{\dot{\boldsymbol{q}}} ||J\dot{\boldsymbol{q}} - \dot{\boldsymbol{X}}||^2 + \lambda||\dot{\boldsymbol{q}}||^2 \tag{B.6}$$

where λ is a constant damping scalar. Eq. (B.6) has the analytic solution

$$\dot{\boldsymbol{q}} = J^T(JJ^T + \lambda^2 I)^{-1}\dot{\boldsymbol{X}} = J^{+,DLS}\dot{\boldsymbol{X}} \tag{B.7}$$

which has the advantage to be always computable and not sensitive to the conditioning of J. The damping constant must be chosen carefully to make Eq. (B.7) numerically stable: the damping constant should be large enough so that the solutions are well-behaved near singularities, but if it is chosen too large, then the errors induced by this regularization may be too large for the considered robotic application. Among methods proposed for selecting damping constants, interested readers can refer to Chiaverini *et al.* (1991), Deo *et al.* (1993) and Hue (2008).

B.3 Exploitation of the redundancy

In case of multiple solutions ($\dot{\boldsymbol{X}} \in Range(J)$ and $Dim(Ker(J)) > 0$), it can be profitable to exploit the possibility to choose a solution adapted to secondary needs.

[1] J is a first order, configuration dependant approximation of the system at the kinematic level. Its validity requires a limitation of the joint space velocities.

B.3.1 Weighted pseudoinverse

The pseudoinverse can be extended to a weighted pseudoinverses ($J^+ \rightarrow J^\#$) when the minimized norms (in joint or operational space) are modified. Again with $J = FG$, we have

$$J^\# = M_q^{-1} F^T [F M_q^{-1} F^T]^{-1} [G^T M_x G]^{-1} G^T M_x \tag{B.8}$$

where M_q is the joint space weighting matrix and M_x is the operational space weighting matrix. M_x and M_q are assumed to be symmetric positive definite.

These weighted pseudoinverses have the following properties:

- Exact and minimal M_q-weighted l_2-norm solution if the equation is solvable (minimization of the norm $\sqrt{\dot{\boldsymbol{q}}^T M_q \dot{\boldsymbol{q}}}$);
- Approximate result with minimal operational M_x-weighted l_2-norm error (minimization of the norm $\sqrt{\dot{\boldsymbol{X}}^T M_x \dot{\boldsymbol{X}}}$) and minimal M_q-weighted l_2-norm solution if the system is not solvable;
- $J^\# = J^{-1}$ if J is square and full rank.

As previously, when the matrix J is not singular, the weighted pseudoinverse has simple expressions:

- if J is full rank in line ($rk(J) = m \leq n$), Eq. (B.8) is equivalent to

$$J^\# = M_q^{-1} J^T (J M_q^{-1} J^T)^{-1} \tag{B.9}$$

- if J_O is full rank in column ($rk(J) = n \leq m$), Eq. (B.8) is equivalent to

$$J^\# = (J^T M_x J)^{-1} J^T M_x. \tag{B.10}$$

Using different weighting matrices is a way to exploit the redundancy of the robot (Chan *et al.* 1995, Park *et al.* 2001, Xiang *et al.* 2010).

B.3.2 Projection on the Jacobian Kernel

The general solution of Eq. (B.1) proposed by Liegeois (1977) is

$$\dot{\boldsymbol{q}} = J^+ \dot{\boldsymbol{X}} + P_J \boldsymbol{z} \tag{B.11}$$

where P_J is a projector on the kernel of J and $\boldsymbol{z}$ an arbitrary vector of $\mathbb{R}^n$. $J^+ \dot{\boldsymbol{X}}$ is the particular solution of Eq. (B.1) while $P_J \boldsymbol{z}$ is its homogeneous solution. When $\boldsymbol{z}$ spans $\mathbb{R}^n$, all the solutions of Eq. (B.1) are reached, which is not the case with weighting techniques as such.

A common and efficient way to compute P_J is given by

$$P_J = (I - J^+ J) \tag{B.12}$$

which yields an orthogonal projector. Using weighted pseudoinverse in Eq. (B.12) is a way to use non orthogonal projectors

$$P_J = (I - J^\# J). \tag{B.13}$$

The projectors relying on expressions involving inversions of J such as Eq. (B.13) and Eq. (B.12) are sensitive to the conditioning of J. In particular, they are subject to the same problem in the neighborhood of singularities. However, using a DLS inverse is not a good way to remedy to this problem. First, the DLS method induces an error in the inversion that distorts the projection. As a consequence, the homogeneous solution term may have a non-homogeneous component, which is not acceptable in the control context as these solutions are used in a context of strict priority. Second, the null space of the Jacobian matrix and a projector onto this null space always exist, even when in singular configuration. Given the Singular Value Decomposition of J

$$J = USV^T \tag{B.14}$$

where U an (m, m)-matrix, S an (m, n)-matrix, and V an (n, n)-matrix, a robust and exact expression of the projector onto the kernel of J is given by

$$P_J = V_{r,n}V_{r,n}^T \tag{B.15}$$

where $V_{r,n}$ is the matrix which content is the r to n columns of V, r being the rank of J (Baerlocher *et al.* 2004).

Usual strict priority multi-objective control laws uses the pseudo-inversion of the matrix $J_2P_{J_1}$ to obtain an optimal solution. For example, in Maciejewski *et al.* (1985)

$$\dot{\boldsymbol{q}} = J_1^+\dot{\boldsymbol{X}}_1 + [J_2(I - J_1^+J_1)]^+(\dot{\boldsymbol{X}}_2 - J_2J_1^+\dot{\boldsymbol{X}}_1) \tag{B.16}$$

Using a DLS pseudo inverse for the term $[J_2(I-J_1^+J_1)]^+$ has the consequence to influence the projection, which may provoke an impact of lower priority objectives on upper priority ones. To circumvent this drawback, a premultiplication by the exact projector must be introduced.

$$\dot{\boldsymbol{q}} = J_1^{+,DLS}\dot{\boldsymbol{X}}_1 + P_{J_1}[J_2P_{J_1}]^{+,DLS}(\dot{\boldsymbol{X}}_2 - J_2J_1^{+,DLS}\dot{\boldsymbol{X}}_1) \tag{B.17}$$

This has the consequence to affect lightly the realization of secondary objective but it enforces the strict priority.

Appendix C

Convex Optimization Methods in IVK problems

This appendix aims to describe briefly the resolution of the Inverse Velocity Kinematics (IVK) problem from a Convex Optimization point of view. Convex Optimization is a domain of non linear programming which proposes very efficient tools to solve a broad class of problems among which IVK in robotics.

C.1 Single hierarchical level: unconstrained problem

C.1.1 Task through equality

The unconstrained problem expressed through equalities is expressed by

$$\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des}(k+1) - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)|| \tag{C.1}$$

where $\dot{\boldsymbol{X}}_{des}(k+1)$ is the current desired operational velocity for the next time step, $\boldsymbol{q}(k)$ is the current configuration, $J_O(\boldsymbol{q}(k))$ is the Jacobian of operational objectives and $\dot{\boldsymbol{q}}(k+1)$ is the joint velocity to find to carry out $\dot{\boldsymbol{X}}_{des}(k+1)$ at next time step. This problem involves a single objective (or a set of objectives at the same hierarchical level, $\dot{\boldsymbol{X}}_{des}(k+1)$ being in that case a concatenation of several operational objectives) and does not consider any constraint. This problem expression means that the aim of the resolution method is to find a control solution $\dot{\boldsymbol{q}}(k+1)$ so that $J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)$ is as close as possible to the desired operational velocity $\dot{\boldsymbol{X}}_{des}$.

C.1.2 Problem resolution

The problem of Eq. (C.1) can be modified slightly without modifying the results (the dependence to $\boldsymbol{q}(k)$ and the time steps k and $(k+1)$ are omitted for the sake of clarity)

$$\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}||^2. \tag{C.2}$$

Considering this problem from the convex optimization point of view means that the problem is solved by evaluating the local behavior of the cost function. In this scope, the first step is to differentiate the cost function with respect to the joint velocity

$$\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}||^2 = 2J_O^T\dot{\boldsymbol{X}}_{des} - 2J_O^TJ_O\dot{\boldsymbol{q}}. \tag{C.3}$$

The optimal value $\dot{\boldsymbol{q}}^*$ is reached when this gradient is null. The particularity of this problem is that this gradient is linear, which enables to use analytic resolutions method; in a more general convex problem, a descent method may have been used. The resolution of the annulation of the gradient is (known as the *optimality condition*[1])

$$J_O^T J_O \dot{\boldsymbol{q}}^* - J_O^T \dot{\boldsymbol{X}}_{des} = 0. \tag{C.4}$$

The kernel of J_O (notation $\mathrm{Ker}(J_O^T)$) and the range of J_O (notation $\mathrm{Rg}(J_O)$) being orthogonal, we have

$$J_O \dot{\boldsymbol{q}}^* - \dot{\boldsymbol{X}}_{des} = 0. \tag{C.5}$$

This result is obvious but it exposes the way quadratic problems are addressed in a convex optimization point of view (1/derivation, 2/descent or use of optimality conditions if possible). This equation is the one addressed by Liegeois (1977).

As seen in appendix B, the general solution of Eq. (C.5) is

$$\dot{\boldsymbol{q}} = J_O^{\#} \dot{\boldsymbol{X}} + P_{J_O} \boldsymbol{z} \tag{C.6}$$

where $J_O^{\#}$ is a weighted pseudoinverse of J_O, P_{J_O} is a projector on $Ker(J_O)$ and $\boldsymbol{z}$ an arbitrary vector of $\mathbb{R}^n$.

C.2 Multiple hierarchical levels: Least Square problem with Equality constraints

The multiple hierarchical levels control is the most largely used control problem formulation. Objectives are hierarchized and the contribution to an objective at a given level cannot be carried out if it impacts higher levels objectives.

C.2.1 Tasks through equalities

A Least Square problem with Equality constraints (LSE) is expressed by

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des}(k+1) - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)|| \\ &\text{subject to} \qquad J_C(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1) - \boldsymbol{b}(k) = 0 \end{aligned} \tag{C.7}$$

where $J_C(\boldsymbol{q}(k))$ is the Jacobian of equality constraints and $\boldsymbol{b}(k)$ its associated offset term. As in the previous section, the dependence to $\boldsymbol{q}(k)$ and the time steps k and $(k+1)$ are omitted.

C.2.2 Optimality conditions

Again, this problem can be written without modification on the results as

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}||^2 \\ &\text{subject to} \quad J_C\dot{\boldsymbol{q}} - \boldsymbol{b} = 0. \end{aligned} \tag{C.8}$$

[1] *Optimality conditions* are equations that enables to check if a solution is optimal with respect to the considered problem. In a few cases (as here), they can be inverted to find directly the optimal solution.

If a solution $\dot{\boldsymbol{q}}_0$ is optimal, then for any other control vector $\dot{\boldsymbol{q}}_1$ satisfying $J_C\dot{\boldsymbol{q}}_1 - \boldsymbol{b} = 0$, the following equation must hold

$$(\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}||^2)^T(\dot{\boldsymbol{q}}_1 - \dot{\boldsymbol{q}}_0) \geq 0. \tag{C.9}$$

This equation can be interpreted as follows: in the kernel of the constraints any displacement from the optimal point has a positive projection on the gradient. Since $\dot{\boldsymbol{q}}_0$ and $\dot{\boldsymbol{q}}_1$ satisfy the constraints, there exists $\dot{\boldsymbol{q}}_{01}$ in $\mathrm{Ker}(J_C)$ such that $\dot{\boldsymbol{q}}_1 = \dot{\boldsymbol{q}}_0 + \dot{\boldsymbol{q}}_{01}$. The optimality condition can then be expressed as

$$(\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}||^2)^T\dot{\boldsymbol{q}}_{01} \geq 0 \quad \forall\dot{\boldsymbol{q}}_{01} \in Ker(J_C). \tag{C.10}$$

As this linear function (left hand term of Eq. (C.10)) is nonnegative on the vector space $\mathrm{Ker}(J_C)$, then it is zero on the subspace. In other words: in the kernel of the constraints any displacement from the optimal point has a null projection on the gradient. As a consequence

$$(\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}_0||^2) \in Ker(J_C)^{\perp} \tag{C.11}$$

and thus

$$(\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}_0||^2) \in Rg(J_C^T). \tag{C.12}$$

Consequently

$$\exists\boldsymbol{\nu} \in \mathbb{R}^p, \quad (\nabla||\dot{\boldsymbol{X}}_{des} - J_O\dot{\boldsymbol{q}}_0||^2)^T + J_C^T\boldsymbol{\nu} = 0. \tag{C.13}$$

where $\boldsymbol{\nu}$ is known as a Lagrangian multipliers vector. Finally, the optimality conditions for a control solution $\dot{\boldsymbol{q}}_0$ are Eq. (C.13) and $J_C\dot{\boldsymbol{q}}_0 - \boldsymbol{b} = 0$. These conditions can be summarized as

$$\begin{bmatrix} J_O^TJ_O & J_C^T \\ J_C & 0 \end{bmatrix}\begin{bmatrix} \dot{\boldsymbol{q}}_0 \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} J_O^T\dot{\boldsymbol{X}} \\ \boldsymbol{b} \end{bmatrix}. \tag{C.14}$$

C.2.3 Analytic resolution through the optimality conditions

The LSE problem can be solved through the resolution of the optimality conditions Eq. (C.14), which comes out to use the well known multi-objective solution (Maciejewski *et al.* 1985)

$$\dot{\boldsymbol{q}} = J_C^{\#}\boldsymbol{b} + (J_O(I - J_C^{\#}J_C))^{\#}(\dot{\boldsymbol{X}} - J_OJ_C^{\#}\boldsymbol{b}) \tag{C.15}$$

when the problem is feasible ($^{\#}$ being the weighted pseudo-inversion operator, see appendix B).

The convex optimization algorithms usually deals with problems with multiple (> 2) hierarchical levels thanks to sequences of quadratic programs. The work of Salini *et al.* (2011) evaluates such a technique.

C.3 Least Square problem with Inequality constraints (LSI)

A Least Square problem with Inequality constraints (LSI) can be expressed by

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}(k+1)\in\mathbb{R}^n} ||\dot{\boldsymbol{X}}_{des}(k+1) - J_O(\boldsymbol{q}(k))\dot{\boldsymbol{q}}(k+1)|| \\ &\text{subject to} \qquad J_C\dot{\boldsymbol{q}}(k+1) - \boldsymbol{b}(k) \leq 0. \end{aligned} \tag{C.16}$$

This problem involves a single objective (or a set of objectives at the same hierarchical level) and some inequality constraints. This problem cannot be treated directly analytically; there is no other way than resorting to iterations to find which constraints are *active* (constraints reached at the optimal solution) and which are not. To avoid iterations, most of the robotics literature techniques take constraints that are fundamentally inequalities (joint physical limits, obstacles) as equalities by introducing avoidance terms. These methods may impose a very large number of conditions, making them potentially not satisfied.

The method exposed to solve such problem is the primal active set method. Active sets methods enable to get information on the constraints that should be activated and/or relaxed in order to find the adequate LSE yielding the optimal value. They are usually composed of 2 phases: phase I is in charge of finding a feasible point, it can be addressed for example through an unconstrained optimization problem; phase II is in charge of minimizing the cost function while maintaining the progression of the solution within the admissibility space. Phase I is not described here, it is assumed that the initial solution $\dot{\boldsymbol{q}}^{(0)}$ satisfies the constraints.

The best way to describe this algorithm is through an example; a more complete approach is given in Nocedal *et al.* (2000).

C.3.1 Example

Fig. C.1 illustrates an inequality constrained quadratic problem.

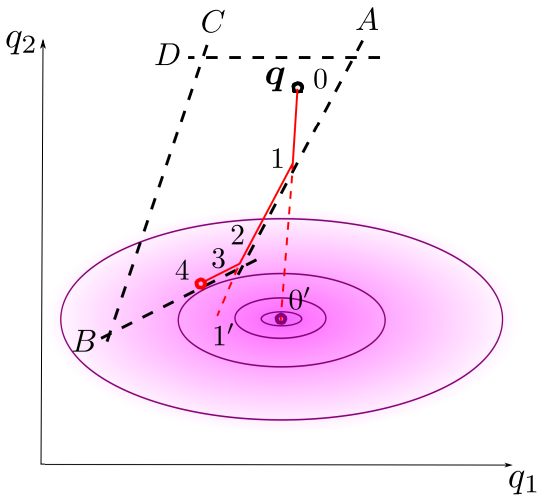

Figure C.1: An example of inequality constrained quadratic problem. A, B, C and D are the linear constraints.

On this example, the initial point is 0, the initial admissible solution of the problem is then $\dot{\boldsymbol{q}}^{(0)}$. The algorithm is incremental; the joint space velocity increments are denoted $\boldsymbol{s}$.

- On the first iteration, without any indication on the constraints to activate, it is

decided to consider that no constraint is active ($J_C^{(0)} = 0$). The quadratic problem

$$\min_{\boldsymbol{s}^{(0)}} ||J(\dot{\boldsymbol{q}}^{(0)} + \boldsymbol{s}^{(0)}) - \dot{\boldsymbol{X}}_{des}|| \quad \text{subject to } J_C^{(0)}\boldsymbol{s}^{(0)} = \boldsymbol{0} \tag{C.17}$$

is solved with $J_C^{(0)} = 0$. This problem is formulated in a particular manner to keep the right hand term null, which enables to avoid to waste an important amount of time when solving the problem thanks to the optimality conditions (see part C.2.3). The obtained solution $\dot{\boldsymbol{q}}^{(0')} = \dot{\boldsymbol{q}}^{(0)} + \boldsymbol{s}^{(0)}$ does not satisfy the admissibility test ($J_C\dot{\boldsymbol{q}}^{(0')} > \boldsymbol{b}$), so $\dot{\boldsymbol{q}}^{(0')}$ is scaled by a factor $u^{(0)} < 1$ (the constraint A is the constraint that limits $u^{(0)}$ here) in order to make the solution compliant with the constraints, thus obtaining the point $\dot{\boldsymbol{q}}^{(1)} = \dot{\boldsymbol{q}}^{(\boldsymbol{0})} + u^{(0)}\boldsymbol{s}^{(0)}$.

- A constraint should be activated to continue the progression. The constraint to add is the one that limited $u^{(0)}$ to the smallest value (constraint A). The problem is then solved another time with the initial point $\dot{\boldsymbol{q}}^{(\boldsymbol{1})}$ and $J_C^{(1)}$ containing uniquely the jacobian (line) associated to constraint A:

$$\min_{\boldsymbol{s}^{(1)}} ||J(\dot{\boldsymbol{q}}^{(1)} + \boldsymbol{s}^{(1)}) - \dot{\boldsymbol{X}}_{des}|| \quad \text{subject to } J_C^{(1)}\boldsymbol{s}^{(1)} = 0. \tag{C.18}$$

The solution found $\dot{\boldsymbol{q}}^{(1')} = \dot{\boldsymbol{q}}^{(1)} + \boldsymbol{s}^{(1)}$ violates a constraint (constraint B). A scaling scalar $u^{(1)}$ is thus used and the solution obtained at this iteration is then $\dot{\boldsymbol{q}}^{(2)} = \dot{\boldsymbol{q}}^{(1)} + u^{(1)}\boldsymbol{s}^{(1)}$.

- $J_C^{(2)}$ contains then 2 active constraints (A and B), and we try to solve the following problem:

$$\min_{\boldsymbol{s}^{(2)}} ||J(\dot{\boldsymbol{q}}^{(2)} + \boldsymbol{s}^{(2)}) - \dot{\boldsymbol{X}}_{des}|| \quad \text{subject to } J_C^{(2)}\boldsymbol{s}^{(2)} = 0. \tag{C.19}$$

The resolution of this problem gives as a solution $\boldsymbol{s}^{(2)} = 0$, because the problem is too much constrained (problem with 2 DOFs constrained by 2 linearly independant equations). The solution $\dot{\boldsymbol{q}}^{(2')}$ is then $\dot{\boldsymbol{q}}^{(2)}$. This solution is admissible (the constraint equation $J_C\dot{\boldsymbol{q}}^{(2')} \leq \boldsymbol{b}$ is satisfied). It is thus kept. In these cases, the optimality conditions are exploited, to know if solution $\dot{\boldsymbol{q}}^{(3)} = \dot{\boldsymbol{q}}^{(2')} = \dot{\boldsymbol{q}}^{(2)}$ is optimal or if a constraint should be removed from the set $J_C^{(2)}$.

$$\begin{bmatrix} J^TJ & (J_C^{(2)})^T \\ J_C^{(2)} & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^{(2')} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -(J^T\boldsymbol{X}_{des})^T \\ \boldsymbol{b}^{(2)} \end{bmatrix} \tag{C.20}$$

Where $\boldsymbol{y}$ is a Lagrange multiplier vector and $\boldsymbol{b}^{(2)}$ the subvector of $\boldsymbol{b}$ containing the components associated to $J_C^{(2)}$. 2 possibilities then:

  - all the components of $\boldsymbol{y}$ are negative of null $\rightarrow \dot{\boldsymbol{q}}^{(2)}$ is the optimal point, which is not the case here.

– at least one of the components of $\boldsymbol{y}$ is positive → the solution can be increased if the associated constraints are removed from $J_C^{(2)}$: our case here for $i = 1$, that is constraint A.

- Constraint A is so removed from $J_C^{(2)}$, yielding matrix $J_C^{(3)}$. The new problem is then addressed:

$$\min_{\boldsymbol{s}^{(3)}} ||J(\dot{\boldsymbol{q}}^{(3)} + \boldsymbol{s}^{(3)}) - \dot{\boldsymbol{X}}_{des}|| \quad \text{subject to } J_C^{(3)}\boldsymbol{s}^{(3)} = 0 \tag{C.21}$$

The resolution gives $\dot{\boldsymbol{q}}^{(3')} = \dot{\boldsymbol{q}}^{(3)} + \boldsymbol{s}^{(3)}$ which is admissible. The optimality condition is then computed:

$$\begin{bmatrix} J^T J & (J_C^{(3)})^T \\ J_C^{(3)} & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^{(3')} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -(J^T \boldsymbol{X}_{des})^T \\ \boldsymbol{b}^{(3)} \end{bmatrix} \tag{C.22}$$

Vector $\boldsymbol{y}$ checks well $\boldsymbol{y} \leq \boldsymbol{0}$, which means that solution $\dot{\boldsymbol{q}}^{(3')}$ is optimal.

C.3.2 Algorithm

The active set method algorithm is then:

C.3.3 Nice guess!

In the case where the optimal active constraints set is known (or guessed) the problem comes out to a unique equalities constrained problem, which can be solved very efficiently. The only extra operation is to check thanks to the optimality conditions that the solution found is the optimal one.

Algorithm 6 : Active set method

0. Initialization of $\dot{\boldsymbol{q}}^{(0)}$ with an admissible solution (satisfying $J_C\dot{\boldsymbol{q}}^{(0)} \leq \boldsymbol{b}$); Initialization of the active constraints matrix $J_C^{(0)}$

1. Resolution of the equalities-constrained problem with a null right hand member.

$$\min_{\boldsymbol{s}^{(k)}} ||J(\dot{\boldsymbol{q}}^{(k)} + \boldsymbol{s}^{(k)}) - \dot{\boldsymbol{X}}_{des}|| \text{ w.r.t. } J_C^{(k)}\boldsymbol{s}^{(k)} = 0 \tag{C.23}$$

by the analytic resolution method based on the optimality conditions (see C.2.3). We obtain the solution $\dot{\boldsymbol{q}}^{(k')} = \dot{\boldsymbol{q}}^{(k)} + \boldsymbol{s}^{(k)}$

2. Admissibility check:

$$J_C\dot{\boldsymbol{q}}^{(k')} \leq \boldsymbol{b} \tag{C.24}$$

If the case, then go to step 4. Else, go to step 3.

3. Computation of the scaling coefficient $u^{(k)}$ such that $\dot{\boldsymbol{q}}^{(k+1)} = \dot{\boldsymbol{q}}^{(k)} + u^{(k)}\boldsymbol{s}^{(k)}$ is admissible and $u^{(k)}$ is maximal. $J_C^{(k)}$ is concatenated with the most constraining constraint (*i.e.* the one that conditions $u^{(k)}$) to yield $J_C^{(k+1)}$; Update: $k := k + 1$; Go back to step 1.

4. Computation of the Lagrange coefficients $\boldsymbol{y}$ in the equation:

$$\begin{bmatrix} J^TJ & (J_C^{(k)})^T \\ J_C^{(k)} & 0 \end{bmatrix} \begin{bmatrix} \dot{\boldsymbol{q}}^{(k')} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} -(J^T\boldsymbol{X}_{des})^T \\ \boldsymbol{b}^{(k)} \end{bmatrix} \tag{C.25}$$

If $\boldsymbol{y} \leq \boldsymbol{0}$, the solution $\dot{\boldsymbol{q}}^{(k')}$ is optimal: get out of the algorithm.
Else, the negative components of $\boldsymbol{y}$ are the constraints that must be removed from $J_C^{(k)}$; Update: $k := k + 1$; Go back to step 1.

Bibliography

[1] Baerlocher, P. and Boulic, R. (2004). An inverse kinematic architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer: International Journal of Computer Graphics*, 20(6):402–417.

[2] Baillieul, J. (1985). Kinematic programming alternatives for redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 722–728.

[3] Bartlett, R., Wachter, A., and Biegler, L. (2000). Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, volume 6, pages 4229–4233.

[4] Bautin, A., Martinez-Gomez, L., and Fraichard, T. (2010). Inevitable collision states: A probabilistic perspective. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4022–4027.

[5] Ben Israel, A. and Greville, T. (2003). *Generalized Inverses: Theory and Applications*. Springer-Verlag, 2nd edition.

[6] Bertolazzi, E., Biral, F., and Lio, M. (2007). Real-time motion planning for multibody systems. *Multibody System Dynamics*, 17(2):119–139.

[7] Biagiotti, L. and Melchiorri, C. (2008). *Trajectory planning for automatic machines and robots*. Springer-Verlag.

[8] Brady, M. (1982). *Robot motion: Planning and control*. The MIT Press.

[9] Brock, O. and Khatib, O. (2000). Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 550–555.

[10] Bruyninckx, H. (2001). Open robot control software: the orocos project. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2523–2528.

[11] Buss, S. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. Unpublished survey article. `http://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf`.

[12] Carbone, G., Ottaviano, E., and Ceccarelli, M. (2007). An optimum design procedure for both serial and parallel manipulators. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 221(7):829–843.

[13] Ceccarelli, M. and Lanni, C. (2004). A multi-objective optimum design of general 3r manipulators for prescribed workspace limits. *Mechanism and Machine Theory*, 39(2):119–132.

[14] Chan, N., Kuffner, J., and Zucker, M. (2008). Improved motion planning speed and safety using regions of inevitable collision. In *Proceedings of the 17th CISM-IFToMM Symposium on Robot Design, Dynamics and Control (RoManSy'08)*.

[15] Chan, T. and Dubey, V. D. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Transactions on Robotics and Automation*, 11(2):286–292.

[16] Chaumette, F. and Marchand, É. (2001). A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing. *IEEE Transactions on Robotics and Automation*, 17(5):719–730.

[17] Chen, I. and Burdick, J. (1995). Determining task optimal modular robot assembly configurations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 132–137.

[18] Cheng, M. and Lin, C. (1997). Genetic algorithm for control design of biped locomotion. *Journal of Robotic Systems*, 14(5):365–373.

[19] Chiaverini, S., Egeland, O., and Kanestrom, R. (1991). Achieving user-defined accuracy with damped least-squares inverse kinematics. In *Proceedings of the Fifth International Conference on Advanced Robotics*, pages 672–677.

[20] Chocron, O. and Bidaud, P. (1997). Genetic design of 3d modular manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 223–228.

[21] Choi, S. and Kim, B. K. (2000). Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, 18(2):143–151.

[22] David, O., Measson, Y., Bidard, C., Rotinat-Libersa, C., and Russotto, F. (2007). Maestro: a hydraulic manipulator for maintenance and decommissioning application. In *Transactions of the European Nuclear Conference*.

[23] David, O., Russotto, F., Da Silva Simoes, M., and Measson, Y. (2011). Interactive anticollision in supervisory control systems for computer aided teleoperation. *Submitted to the Robotics and Automation Magazine*.

[24] Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley and Sons, Inc.

[25] Deb, K., Pratap, A., Agrawal, S., and Meyrivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[26] Decre, W., Smits, R., Bruyninckx, H., and De Schutter, J. (2009). Extending itasc to support inequality constraints and non-instantaneous task specification. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 964–971.

[27] Deo, A. and Walker, I. (1993). Adaptive non-linear least squares for inverse kinematics. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 186–193.

[28] Doty, K., Melchiorri, C., and Bonivento, C. (1993). A theory of generalized inverses applied to Robotics. *The International Journal of Robotics Research*, 12(1):1–19.

[29] Ehmann, S. and Lin, M. (2001). Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Computer Graphics Forum*, volume 20, pages 500–511.

[30] Escande, A., Mansard, N., and Wieber, P.-B. (2010). Fast resolution of hierarchized inverse kinematics with inequality constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3733–3738.

[31] Faverjon, B. and Tournassoud, P. (1987). A local based approach for path planning of manipulators with a high number of degrees of freedom. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1152–1159.

[32] Fonseca, C. and Fleming, P. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423.

[33] Fox, D., Burgard, W., and Thrun, S. (1997). The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33.

[34] Fraichard, T. (2007). A short paper about motion safety. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1140–1145.

[35] Fraichard, T. and Asama, H. (2004). Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024.

[36] Gallant, M. and Boudreau, R. (2002). The synthesis of planar parallel manipulators with prismatic joints for an optimal, singularity-free workspace. *Journal of Robotic Systems*, 19(1):13–24.

[37] Gicquel, P., Andriot, C., Lauture, F., Measson, Y., and Desbats, P. (2001). Tao2000: a generic control architecture for advanced computer aided teleoperation systems. In *Proceedings of the 9th ANS Topical Meeting on Robotics and Remote Systems*.

[38] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition.

[39] Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33.

[40] Guilbert, M., Joly, L., and Wieber, P.-B. (2008). Optimization of complex robot applications under real physical limitations. *The International Journal of Robotics Research*, 27(5):629–644.

[41] Haddadin, S., Albu-Scheffer, A., Eiberger, O., and Hirzinger, G. (2010). New insights concerning intrinsic joint elasticity for safety. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 2181–2187.

[42] Han, J., W.K., C., Youm, Y., and Kim, S. (1997). Task based design of modular robot manipulator using efficient genetic algorithm. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 507–512.

[43] Hauser, K. (2011). *Algorithmic Foundations of Robotics IX*, volume 68 of *Springer Tracts in Advanced Robotic*, chapter Adaptive Time Stepping in Real-Time Motion Planning, pages 139–155. Springer.

[44] Herrb, M. (2010). Gdhe-graphic display for hilare experiments version 3.8. User Guide.
`ftp://softs.laas.fr/pub/openrobots/gdhe/gdhe_eng.pdf`.

[45] Holland, J. (1992). *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA.

[46] Horn, J., Nafploitis, N., and Goldberg, D. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE Conference on Evolutionary Computation*, volume 1, pages 82–87.

[47] Hue, V. (2008). *Simulation de mouvement humain sur postes de travail pour le diagnostic et l'aide à la conception*. PhD thesis, Laboratoire Génie de Production, Institut National Polytechnique de Toulouse.

[48] Huo, L. and Baron, L. (2011). The self-adaptation of weights for joint-limits and singularity avoidances of functionally redundant robotic-task. *Robotics and Computer-Integrated Manufacturing*, 27(1):367–376.

[49] Ikuta, K., Ishii, H., and Nokata, M. (2003). Safety evaluation method of design and control for human-care robots. *The International Journal of Robotics Research*, 22(5):281–297.

[50] Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E., and Laumond, J.-P. (2008). A local collision avoidance method for non-strictly convex polyhedra. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland.

[51] Kanoun, O., Lamiraux, F., Wieber, P., Kanehiro, F., Yoshida, E., and Laumond, J. (2009). Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2939–2944.

[52] Keith, F., Mansard, N., Miossec, S., and Kheddar, A. (2009). Optimization of tasks warping and scheduling for smooth sequencing of robotic actions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1609–1614.

[53] Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98.

[54] Khatib, O., Yokoi, K., Brock, O., Chang, K., and Casal, A. (2001). Robots in human environments. *Archives of Control Sciences, Special Issue on Recent Developments in Robotics, part I.*, 11(3–4):123–138.

[55] Kim, J. and Khosla, P. (1992). A multi-population genetic algorithm and its application to design of manipulators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 279–286.

[56] Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems*, volume 58 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Heidelberg, Germany.

[57] Kulic, D. and Croft, E. (2006). Real-time safety for human-robot interaction. *Robotics and Autonomous Systems*, 54(1):1–12.

[58] Le Pechon, J.-C., Sterk, W., and Van Rees Vellinga, T. (2000). Saturation diving for tunnelling operations. In *Proceedings of the XXVI Annual Scientific Meeting of European Underwater and Baromedical Society*, page 274.

[59] Lee, E. and Mavroidis, C. (2004). Geometric design of 3r robot manipulators for reaching four end-effector spatial poses. *The International Journal of Robotics Research*, 23(3):247–254.

[60] Leger, P. (1999). *Automated synthesis and Optimization of robot configuration: an evolutionary approach*. PhD thesis, Carnegie Mellon University.

[61] Liegeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(12):868–871.

[62] Maciejewski, A. and Klein, C. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117.

[63] Mansard, N. and Chaumette, F. (2007). Task sequencing for high-level sensor-based control. *IEEE Transactions on Robotics*, 23(1):60–72.

[64] Mansard, N. and Chaumette, F. (2009). Directional redundancy for robot control. *IEEE Transactions on Automatic Control*, 54(6):1179–1192.

[65] Mansard, N., Khatib, O., and Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, 25(3):670–685.

[66] Martinez-Gomez, L. and Fraichard, T. (2008). An efficient and generic 2d inevitable collision state-checker. In *Proceedings of the IEEE-RSJ International Conference on Intelligent Robots and Systems*, pages 234–241.

[67] Martinez-Gomez, L. and Fraichard, T. (2009). Collision avoidance in dynamic environments: an ics-based solution and its comparative evaluation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 100–105.

[68] Mouret, J.-B. and Doncieux, S. (2010). Sferes_v2: Evolvin' in the multi-core world. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1–8.

[69] Nakamura, Y. and Hanafusa, H. (1986). Inverse kinematics solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement and Control*, 108(3):163–171.

[70] Nocedal, J. and Wright, S. (2000). *Numerical optimization*. Springer-Verlag, 1st edition.

[71] Padois, V. (2005). *Enchaînements dynamiques de tâches pour des manipulateurs mobiles a roues*. PhD thesis, Laboratoire Génie de Production, Institut National Polytechnique de Toulouse.

[72] Padois, V., Fourquet, J., and Chiron, P. (2007). Kinematic and dynamic model-based control of wheeled mobile manipulators: A unified framework for reactive approaches. *Robotica*, 25(2):157–173.

[73] Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H. (2007). *Engineering Design: A Systematic Approach*. Springer, 3rd edition.

[74] Palmer, C. and Kershenbaum, A. (1995). An approach to a problem in network design using genetic algorithms. *Networks*, 26(3):151–163.

[75] Park, J., Choi, Y., Chung, W. K., and Youm, Y. (2001). Multiple tasks kinematics using weighted pseudo-inverse for kinematically redundant manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 4, pages 4041–4047.

[76] Park, J. and Khatib, O. (2008). Robot multiple contact control. *Robotica*, 26(5):667–677.

[77] Parthasarathi, R. and Fraichard, T. (2007). An inevitable collision state-checker for a car-like vehicle. In *Proceedings of the IEEE International Conference On Robotics and Automation*, pages 3068–3073.

[78] Peinado, M., Meziat, D., Maupu, D., Raunhardt, D., Thalmann, D., and Boulic, R. (2009). Full-body avatar control with environment awareness. *Computer Graphics and Applications*, 29(3):62–75.

[79] Petti, S. and Fraichard, T. (2005). Safe motion planning in dynamic environments. In *Proceedings of th IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2210–2215.

[80] Purshouse, R. and Fleming, P. (2003). Conflict, harmony and independence: Relationships in evolutionary multi-criterion optimisation. In *Proceedings of the International Conference on Evolutionary Multi-criterion Optimization*, volume 2632, pages 16–30.

[81] Raunhardt, D. and Boulic, R. (2007). Progressive clamping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4414–4419.

[82] Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms*. Springer, 2nd edition.

[83] Rubrecht, S., Padois, V., and Bidaud, P. (2009). *New Horizons in Evolutionary Robotics: Extended Contributions from the 2009 EvoDeRob Workshop*, volume 341 of *Studies in Computational Intelligence*, chapter Evolutionary Design of a Robotic Manipulator for a Highly Constrained Environment, pages 59–64. Springer.

[84] Rubrecht, S., Padois, V., Bidaud, P., and Broissia, M. (2010a). *Advances in Robot Kinematics: Motion in Man and Machine*, chapter Constraint compliant control for a redundant manipulator in a cluttered environment, pages 367–376. Springer.

[85] Rubrecht, S., Padois, V., Bidaud, P., and De Broissia, M. (2010b). Constraints compliant control: constraints compatibility and the displaced configuration approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 677–684.

[86] Rubrecht, S., Padois, V., Bidaud, P., de Broissia, M., and Da Silva Simoes, M. (2011). Motion safety and constraints compatibility for multibody robots. *Revision submitted – Autonomous Robots*.

[87] Salini, J., Padois, V., and Bidaud, P. (2011). Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1283–1290.

[88] Salle, D., Bidaud, P., and Morel, G. (2004). Optimal design of high dexterity modular mis instrument for coronary artery bypass grafting. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1276–1281.

[89] Sentis, L. and Khatib, O. (2005). Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *The International Journal of Humanoid Robotics*, 2(4):505–518.

[90] Seraji, H. (1989). Configuration control of redundant manipulators: Theory and implementation. *IEEE Transactions on Robotics and Automation*, 5(4):472–490.

[91] Seraji, H. and Colbaugh, R. (1990). Improved configuration control for redundant robots. *Journal of Robotic Systems*, 7(6):897–928.

[92] Shen, W. and Gu, J. (2007). Multi-criteria kinematics control for the pa10-7c robot arm with robust singularities. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, pages 1242–1248.

[93] Siciliano, B. and Slotine, J.-J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. In *Proceedings of the IEEE International Conference on Advanced Robotics*, volume 2, pages 1211–1216.

[94] Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st Conference on Computer Graphics and Interactive Techniques*, pages 15–22.

[95] Singla, E., Tripathi, S., Rakesh, V., and Dasgupta, B. (2010). Dimensional synthesis of kinematically redundant serial manipulators for cluttered environments. *Robotics and Autonomous Systems*, 58(5):585–595.

[96] Sotzing, C., Htay, W., and Congdon, C. (2005). Gencem: A genetic algorithms approach to coordinated exploration and mapping with multiple autonomous robots. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 3, pages 2317–2324.

[97] Srinivas, M. and Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):656–667.

[98] Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

[99] Van Den Berg, J., Ferguson, D., and Kuffner, J. (2006). Anytime path planning and replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2366–2371.

[100] Wampler, C. (1986). Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):93–101.

[101] Xiang, J., Zhong, C., and Wei, W. (2010). General-weighted least-norm control for redundant manipulators. *IEEE Transactions on Robotics*, 26:660–669.

[102] Zinn, M., Khatib, O., Roth, B., and Salisbury, J. (2004). Playing it safe [human-friendly robots]. *IEEE Robotics and Automation Magazine*, 11(2):12–21.

[103] Zitzler, E. and Thiele, L. (1998). *Parallel Problem Solving from Nature*, volume 1498 of *Lecture Notes in Computer Science*, chapter Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study, pages 292–301. Springer.