HAL
open science

Analysis of the Discontinuities in Prioritized Tasks-Space Control Under Discrete Task Scheduling Operations

François Keith, Pierre-Brice Wieber, Nicolas Mansard, Abderrahmane Kheddar

To cite this version:

François Keith, Pierre-Brice Wieber, Nicolas Mansard, Abderrahmane Kheddar. Analysis of the Discontinuities in Prioritized Tasks-Space Control Under Discrete Task Scheduling Operations. IROS'11: International Conference on Intelligent Robots and Systems, Sep 2011, San Francisco, United States. pp.3887-3892, 10.1109/IROS.2011.6094706 . hal-00649287

HAL Id: hal-00649287

https://inria.hal.science/hal-00649287

Submitted on 7 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Analysis of the Discontinuities in Prioritized Tasks-Space Control Under Discreet Task Scheduling Operations

François Keith Pierre-Brice Wieber Nicolas Mansard Abderrahmane Kheddar

***Abstract*— This paper examines the control continuity in hierarchical task-space controllers. While the continuity is ensured for any a priori fixed number of tasks –even in ill-conditioned configurations–, the control resulting from a hierarchical stack-of-task computation may not be continuous under some discrete events. In particular, we study how the continuity of the stack-of-task control computation is affected under discreet scheduling operations such as on-the-fly priority switching between tasks, or tasks insertion and removal, which changes the number of tasks in the stack controller. Different ways to formulate a hierarchy of tasks are presented together with their continuity properties, which is thoroughly analyzed under such discreet scheduling operations.**

I. Introduction

The task function approach [15], [8] is a generic approach to produce intuitively sensor-based robot objectives. These tasks only define a motion on part of the robot, hence it is possible to take advantage of the redundancy of the system in order to realize several tasks simultaneously. A classical approach is to organize them into a hierarchy [13], [17], [10], which enables to build complex behavior for very redundant robot such as humanoids [1], [16], [12].

Fig. 1 grossly illustrates our approach in using the task-space as a basic component to plan and program complex robot behaviors, ranging from and bridging high level task-based planning to low-level control [12], [7]. In this architecture several levels of reactivity are needed. The inner one is the classical low-level control loop, which is 'encapsulated' in the task-space control loop using the hierarchical stack of tasks as the control computation. We also add a higher loop that feeds back the robot and environment state, problems, changes... to the task planner which role is to select and schedule the tasks according to such events. Such a high-level loop is even unavoidable in highly varying environments. Our focus in this paper is to embed the task scheduler with the capability to act on the fly on the stack-of-task controller using discreet operations such as tasks insertion, removal and priority switching. In particular, we address the fundamental problem of the controller continuity under such discreet operations, which is also linked to the structural algorithmic computations of the stack-of-task.

There are different ways to define a hierarchy between the tasks. In [3], the priority between the tasks is defined by weighting the influence of each task with respect to the others. In [17], [6], [5], a stack-of-task mechanism is defined,

F. Keith and P.-B. Wieber are with the INRIA Rhône-Alpes, Grenoble, France; N. Mansard is with the CNRS-LAAS, Toulouse, France; and A. Kheddar is with the CNRS-UM2 LIRMM, Montpellier, France and CNRS-AIST JRL, UMI3218/CRT, Tsukuba, Japan

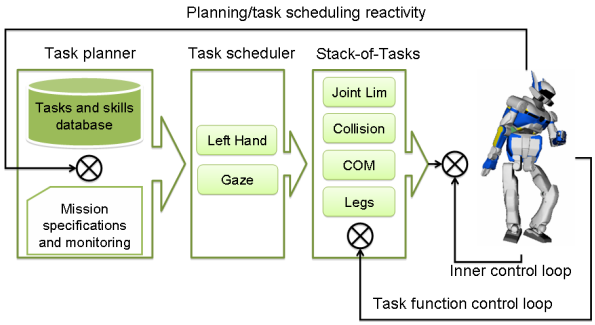

Fig. 1. Task-space based architecture bridging task planning to low-level closed-loop control.

where the priority between the tasks is ensured by realizing each task in the null space left by tasks of higher priority. For a fixed set of tasks, these formulations ensure the continuity of control law even in ill-conditioned cases [2].

Yet, to our best knowledge, the problem of the continuity subsequent to discreet scheduling operations on hierarchical task space controllers was not investigated. To this purpose, using the weight-based method is the easiest solution, since the solution consists in realizing the swap numerically by modifying the weight of the appropriate tasks [14]. However, all possible tasks must be considered a priori; hence we expect problems if the number of tasks changes on the fly.

Since we are using the stack-of-task method, we investigate the continuity property only for this scheme. Recently, a method has been proposed to smooth the realization of discrete events in a stack of tasks [9], consisting in defining an intermediate value during the transition period. Instead, we focus on the influence of the damping during such a swap process. The damping is a classical solution used when the control is ill-conditioned (e.g. when the tasks are conflicting).

We discuss a method which guarantees the continuity of the control (with respect to the time), and consider its impact on the hierarchy between tasks. Section II recalls the formalisms used in the literacy to realize a hierarchy of tasks and how they handle singular cases, while Section III illustrates the potential discontinuities. In Sections IV and V, we tackle the issue of the continuity during the realization of discrete events: the swap of consecutive tasks and the insertion/removal of tasks. Finally, Section VI exemplifies the models presented on the HRP-2 robot.

II. Structure of the Hierarchical Stack of Tasks

We recall the two main solutions to compute the control law of a stack of tasks. Both formulations are discontinuous at sequencing points. They will then be reformulated in the following section, to ensure the continuity.

We consider a robot with k degrees of freedom, and note $\mathbf{q}$ its configuration vector ($k = \dim(\mathbf{q})$). In the following, the robot input control is the joint velocity $\dot{\mathbf{q}}$.

A. Regulation of one task

A task is defined by three elements: a vector $\mathbf{e}$, a Jacobian $\mathbf{J}$ and a reference evolution of the task function $\dot{\mathbf{e}}^*$. It is noted $\mathbf{e}$ or, when more details are needed, $(\mathbf{e}, \mathbf{J}, \dot{\mathbf{e}}^*)$. Typically, the vector $\mathbf{e}$ corresponds to the error between a signal $\mathbf{s}$ and its desired value $\mathbf{s}^*$: $\mathbf{e} = \mathbf{s} - \mathbf{s}^*$. The Jacobian $\mathbf{J}$ binds the error and the vector $\mathbf{q}$ according to the equation $\mathbf{J} = \frac{\partial e}{\partial q}$. The direct control equation is thus:

$$\dot{\mathbf{e}} = \mathbf{J}\dot{\mathbf{q}} \tag{1}$$

The reference behavior $\dot{\mathbf{e}}^*$ defines the way the error is handled. For example, the regulation of the error can be imposed as an exponential decrease by using $\dot{\mathbf{e}}^* = -\lambda\mathbf{e}, \lambda \in \mathbb{R}^+$. The control value giving the best regulation of a single task is found by solving the following minimization problem:

$$\min_{\dot{\mathbf{q}} \in \mathbb{R}^k} \frac{1}{2}\|\mathbf{J}\dot{\mathbf{q}} - \dot{\mathbf{e}}^*\|^2 \tag{2}$$

This can also be formulated as a least square solution:

$$\dot{\mathbf{q}} = \mathbf{J}^+\dot{\mathbf{e}}^* + \mathbf{P}\mathbf{z} \tag{3}$$

where $\mathbf{J}^+$ is the least square of $\mathbf{J}$, and $\mathbf{P} = \mathbf{I} - \mathbf{J}^+\mathbf{J}$ the projector into the null space of $\mathbf{J}$. While the term $\mathbf{J}^+\dot{\mathbf{e}}^*$ ensures the regulation of the task, the term $\mathbf{P}\mathbf{z}$ corresponds to an additional control that does not modify the regulation of the task. If $\mathbf{z} = \mathbf{0}$, $\dot{\mathbf{q}}$ corresponds to the least square solution.

B. Regulation of n tasks

Generally, the task function $\mathbf{e}$ only defines a control law for a subpart of the robot. It is hence possible to realize several tasks simultaneously. We note $(\mathbf{e_i}, \mathbf{J_i}, \dot{\mathbf{e}}_\mathbf{i}^*)$ the i-th task and $\dot{\mathbf{q}}_\mathbf{i}$ the control law associated to the tasks $\mathbf{e_1}, \ldots, \mathbf{e_i}$.

1) Pseudo inverse: In order to realize a hierarchy among a set of n tasks, each task (except the first one) is performed in the null space of higher priority ones. The resolution algorithm, detailed in [6], consists in solving for each task a minimization problem, such as:

$$\min_{\dot{\mathbf{q}}_\mathbf{i} \in S_i} \frac{1}{2}\|\mathbf{J_i}\dot{\mathbf{q}}_\mathbf{i} - \dot{\mathbf{e}}_\mathbf{i}^*\|^2 \tag{4}$$

where S_i the null space left by the tasks $\mathbf{e_1}, \ldots, \mathbf{e_{i-1}}$: $S_1 = \mathbb{R}^k$ and $S_i = S_{i-1} \cap \text{null}(\mathbf{J_{i-1}}), 1 < i \leq n$. By limiting the choice of $\dot{\mathbf{q}}_\mathbf{i}$ to S_i, the lower priority tasks do not affect the execution of higher priority ones.

The control law for a task $\mathbf{e_i}$ is given in [17]

$$\dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_\mathbf{i-1} + (\mathbf{J_i}\mathbf{P_{i-1}})^+(\dot{\mathbf{e}}_\mathbf{i}^* - \mathbf{J_i}\dot{\mathbf{q}}_\mathbf{i-1}) \tag{5}$$

$$\text{where} \quad \mathbf{P_i} = \mathbf{I} - \begin{bmatrix} \mathbf{J_1} \\ \vdots \\ \mathbf{J_i} \end{bmatrix}^+ \begin{bmatrix} \mathbf{J_1} \\ \vdots \\ \mathbf{J_i} \end{bmatrix}$$

For a stack of tasks containing n tasks, the sequence starts with $\dot{\mathbf{q}}_\mathbf{0} = \mathbf{0}$ and the final control value is $\dot{\mathbf{q}} = \dot{\mathbf{q}}_\mathbf{n}$ (see [10] for more details). The use of a pseudo-inverse implicitly ensures that the solution is of minimum norm. This minimum norm is not explicitly imposed by the quadratic-problem formulation (4), but is just a pleasant side-effect of the pseudo-inverse properties. Therefore, the obtained control law is equivalent to the control law from the same stack of task, plus the final quadratic program:

$$\min_{\dot{\mathbf{q}}_\mathbf{n+1} \in S_{n+1}} \frac{1}{2}\|\dot{\mathbf{q}}_\mathbf{n+1}\|^2 \tag{6}$$

It is as if the stack was always augmented with the final task $(\mathbf{e_{n+1}}, \mathbf{I}, \mathbf{0})$ (identity Jacobian, zero reference), that explicitly ensures the minimum norm of the solution. Similarly, a minimum-weighted norm can be imposed, that generalizes the use of weighted pseudo inverse [4]. This task is not necessary, it simply explicits the properties of the pseudo-inverse. This remark is used in the next section to build a continuous control law.

Assuming that for each task $\dot{\mathbf{e}}_\mathbf{i}^*$ is continuous and that $\mathbf{J_i}$ has constant rank, this formulation leads to a continuous evolution of the control. Break of discontinuities due to rank changes of $\mathbf{J_i}$ is discussed in the following paragraph. However, nothing ensures that the continuity holds if the number of tasks changes.

2) Damping: The methods (4) and (5) correspond to the ideal implementations of the stack of tasks and ensure a continuous evolution of the control as far as there are no kinematic and algorithmic singularities. In practice, using a pseudo-inverse is too hazardous: when approaching the singularities, the inverse of the Jacobian reaches excessive values, causing immoderate control values. A well-known solution consists in modifying the behavior for very small singular values. The method (4) is adapted as follows:

$$\min_{\dot{\mathbf{q}}_\mathbf{i} \in S_i} \frac{1}{2}\|\mathbf{J_i}\dot{\mathbf{q}}_\mathbf{i} - \dot{\mathbf{e}}_\mathbf{i}^*\|^2 + \frac{1}{2}\delta\|\dot{\mathbf{q}}_\mathbf{i}\|^2 \tag{7}$$

where δ a real positive constant. Similarly, a damped inversion, defined by $\mathbf{M}^\dagger = (\mathbf{M} + \delta\mathbf{I})^+$, can be realized instead of the pseudo-inverse, and gives the solution of the damped quadratic program. The equation (5) is modified as follows:

$$\dot{\mathbf{q}}_\mathbf{i} = \dot{\mathbf{q}}_\mathbf{i-1} + (\mathbf{J_i}\mathbf{P_{i-1}})^\dagger(\dot{\mathbf{e}}_\mathbf{i}^* - \mathbf{J_i}\dot{\mathbf{q}}_\mathbf{i-1}) \tag{8}$$

This method preserves the continuous evolution of the control when crossing a singularity, in particular an *algorithmic* singularity [2] when the tasks become incompatible. However, introducing a damping factor weakens the hierarchy between tasks: the higher the threshold is, the weaker the priority order will be. Moreover, the damping factor reduces the accuracy of the algorithm resolution, for all the cases, poorly and properly conditioned. In consequence, the task references $\dot{\mathbf{e}}_\mathbf{i}^*$ are not properly achieved, and the stack hierarchy is weakened. The accuracy is more diminished when the damping factor increases. The tuning of this factor is then difficult, requiring a trade-off between accuracy and robustness to singularity [2]. In practice, a same reasonable value can be applied for a large panel of problems.

C. The hierarchy of tasks as a limit of the weighting process

The control law can also be formulated as the solution of a single minimization problem for the entire stack. The influence of each task can then be determined by associating to each task an arbitrary weight. A kind of hierarchy between the tasks is realized by associating to low priority tasks a small weight so that their impact on $\dot{\mathbf{q}}$ is negligible compared to higher priority ones (which results as only one task active).

1) Two tasks: For a stack containing two tasks $\mathbf{e_1}$ and $\mathbf{e_2}$, the control law is:

$$\lim_{\epsilon \to 0} \left(\min_{\dot{\mathbf{q}}} \|\mathbf{J_1}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{1}^*\|^2 + \epsilon \|\mathbf{J_2}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{2}^*\|^2 \right) \tag{9}$$

The priority of the task $\mathbf{e_1}$ on $\mathbf{e_2}$ is ensured by the factor ϵ: if it is small enough, the system behaves similarly to a classical hierarchy of tasks. Note that this equation does not define the control law when ϵ *equals* 0, since the corresponding value would be $\dot{\mathbf{q}} = \mathbf{J}_\mathbf{1}^+ \dot{\mathbf{e}}_\mathbf{1}^*$.

The strict hierarchy can then be obtained by pushing the ϵ parameter to the 0 limit. Formally, the result can be written as follow. Let the matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times n}$, such that $\text{rank}(\mathbf{B}) = p$, and the vectors $\mathbf{a} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^p$ and $\mathbf{x} \in \mathbb{R}^n$. The Least Square Estimate problem with constraints:

$$\mathbf{x}_{\text{LSE}}^* = \begin{cases} \min\limits_{\mathbf{x}} \|\mathbf{B}\mathbf{x} - \mathbf{b}\|^2 \\ st.\ \mathbf{A}\mathbf{x} = \mathbf{a} \end{cases} \tag{10}$$

and the least square problem without constraints where one of the equation is balanced by the coefficient $\mu > 0$:

$$\mathbf{x}^*(\mu) = \min_{\mathbf{x}_\mu} \left\| \begin{pmatrix} \mu\mathbf{A} \\ \mathbf{B} \end{pmatrix} \mathbf{x}_\mu - \begin{pmatrix} \mu\mathbf{a} \\ \mathbf{b} \end{pmatrix} \right\|^2 \tag{11}$$

are such that $\lim\limits_{\mu \to \infty} \mathbf{x}^*(\mu) = \mathbf{x}_{\text{LSE}}^*$. The proof of the equivalence is given in [18].

2) n task: Extending (9) to n tasks is straightforward:

$$\lim_{\epsilon \to 0} \min_{\dot{\mathbf{q}}} \left(\sum_{i=1}^{n} \left(\|\mathbf{J_i}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{i}^*\|^2 \epsilon^{(i-1)} \right) + \|\dot{\mathbf{q}}\|^2 \epsilon^n \right) \tag{12}$$

In practice, the value of the coefficient ϵ is user-defined. When there is a large number of tasks, choosing the value of ϵ is difficult: for a too high value, the hierarchy between the tasks is only approximate, whereas for a too small value, the dependency in ϵ^n may cause numerical issue, and the lower priority tasks may not be taken into account any more.

The presented methods detail the computation of the control for a given set of tasks of fixed order, and how the continuity of the control law can be ensured. In practice, it is most likely that one requires the realization of a sequence of tasks, not only a given set of hierarchical tasks, which implies the addition of discrete operations (task swap, insertion...).

III. Discrete-event related discontinuities

Two possible causes of discontinuity are considered under discrete task operation. The first one corresponds to the additional control value: the insertion of a task in an empty stack, switching from a null control to a non-null control, is a trivial example. The second one is the modification of the null space associated to a task, that may occur during the insertion or removal of a task anywhere other than at the end of the stack, or during the swap of two tasks. This can be illustrated by considering a stack of tasks with only two incompatible tasks that will be swapped, $(\mathbf{e_1}) \begin{cases} x = 1 \\ y = 1 \end{cases}$ and $(\mathbf{e_2}) \begin{cases} x = 2 \\ z = 2 \end{cases}$.

Because of the incompatibility of the tasks, two different behaviours are possible depending on their respective priority: $\dot{\mathbf{q}}_{[\mathbf{1}|\mathbf{2}]} = \begin{bmatrix} 1 & 1 & 2 \end{bmatrix}^T$ and $\dot{\mathbf{q}}_{[\mathbf{2}|\mathbf{1}]} = \begin{bmatrix} 2 & 1 & 2 \end{bmatrix}^T$ where $\dot{\mathbf{q}}_{[\mathbf{A}|\mathbf{B}]}$ is the control law corresponding to the stack of tasks where $\mathbf{e_A}$ has priority on $\mathbf{e_B}$. Realizing an instantaneous swap will result in a punctual discontinuity in the control shape. A smoothing of the control law is required, and the smoothing period is noted $[t_s^I, t_s^F]$.

IV. Smooth control while swapping two tasks

First, we show that the swap of two incompatible tasks $\mathbf{e_1}$ and $\mathbf{e_2}$ cannot be written as a minimization problem, then we propose a method to realize it and extend it to n tasks.

A. Single minimization problem formulation

1) Ideal case: The swap is realized by temporarily placing the two tasks at the same level and modifying the weight of each task. During the swap, the control law is:

$$\min_{\dot{\mathbf{q}} \in S_1} \frac{1-\alpha}{2} \|\mathbf{J_1}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{1}^*\|^2 + \frac{\alpha}{2} \|\mathbf{J_2}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{2}^*\|^2 \tag{13}$$

where α is the swap coefficient depending on time[1] defined during the smoothing period:

$$\begin{cases} \alpha(t) : [t_s^I, t_s^F] \to)0, 1(\\ \lim\limits_{t \to t_s^I} \alpha(t) = 0 \quad \text{and} \quad \lim\limits_{t \to t_s^F} \alpha(t) = 1 \end{cases} \tag{14}$$

The control law corresponding to (13) is defined by:

$$\begin{aligned} \dot{\mathbf{q}} &= \left(\mathbf{H} \begin{bmatrix} \mathbf{J_1} \\ \mathbf{J_2} \end{bmatrix} \right)^+ \left(\mathbf{H} \begin{bmatrix} \dot{\mathbf{e}}_\mathbf{1}^* \\ \dot{\mathbf{e}}_\mathbf{2}^* \end{bmatrix} \right) \\ \text{where} \quad \mathbf{H}(\alpha) &= \begin{bmatrix} (1-\alpha)\mathbf{I_1} & 0 \\ 0 & \alpha\mathbf{I_2} \end{bmatrix} \end{aligned} \tag{15}$$

The matrix $\mathbf{H}$ can be interpreted as an activation matrix [10]. Eq. (15) is continuous for $\alpha \in)0, 1($: $\mathbf{HJ}$ and $\mathbf{He}$ are continuous, $\mathbf{HJ}$ has a constant rank and the inversion of a matrix is a continuous operation when the rank of the matrix is constant. At the limits $t \to t_s^I$ and $t \to t_s^F$, the control is equivalent to the one presented in Section II-C.1. This equivalence ensures the continuity with the strict control law for the two orders $(\mathbf{e_1}, \mathbf{e_2})$ and $(\mathbf{e_2}, \mathbf{e_1})$.

2) Facing singularities: The control $\dot{\mathbf{q}} = (\mathbf{HJ})^+(\mathbf{H}\dot{\mathbf{e}}^*)$, where $\mathbf{H}$ is an activation matrix, is known to present discontinuities in case of rank change [11]. Hence, when the matrix $[\mathbf{J_1}, \mathbf{J_2}]$ presents an algorithmic singularity (i.e. $\mathbf{rank}([\mathbf{J_1}, \mathbf{J_2}]) < \mathbf{rank}(\mathbf{J_1}) + \mathbf{rank}(\mathbf{J_2})$), the method (15) introduces discontinuities during the swap.

These discontinuities cannot be solved using the classic damping method: $\dot{\mathbf{q}} = (\mathbf{HJ})^\dagger(\mathbf{H}\dot{\mathbf{e}}^*)$. Indeed, damping the

[1] To simplify the notation, the dependency in t is omitted in next equations

pseudo-inverse relative to the Jacobians leads to a discontinuous evolution of the control even in the compatible case. This problem appears when the term α becomes negligible compared to the damping factor δ. The second task is then shadowed and the control is closer to the one where only one task is in the stack. To highlight this, let us formulate the minimization problem of the damped control law during the swap. Eq. (15) is the solution of:

$$\min_{\dot{\mathbf{q}} \in S_1} \frac{1-\alpha}{2} \|\mathbf{J_1}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{1}^*\|^2 + \frac{\alpha}{2}\|\mathbf{J_2}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{2}^*\|^2 + \delta\|\dot{\mathbf{q}}\|^2 \quad (16)$$

where $\delta\|\dot{\mathbf{q}}\|^2$ is the term qualifying the damped inverse, that can be seen as a third task $(\mathbf{e}_\delta, \mathbf{I}, \mathbf{0})$ of weight δ. Suppose that $\alpha \ll \delta \ll 1$, the minimization problem can write as:

$$\min_{\dot{\mathbf{q}}} \left(\beta_1\|\mathbf{J_1}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{1}^*\|^2 + \beta_2\|\dot{\mathbf{q}}\|^2 + \beta_3\|\mathbf{J_2}\dot{\mathbf{q}} - \dot{\mathbf{e}}_\mathbf{2}^*\|^2\right) \quad (17)$$

where $\beta_1 = \frac{1-\alpha}{2}$, $\beta_2 = \delta$, $\beta_3 = \frac{\alpha}{2}$, and $\beta_1 \gg \beta_2 \gg \beta_3$. Intuitively, this corresponds to the control law associated to a stack containing the three tasks with order $\mathbf{e_1}$, $\mathbf{e}_\delta$, $\mathbf{e_2}$. The null space left to realize the task $\mathbf{e_2}$ is empty, preventing its regulation: the stack acts as if only $\mathbf{e_1}$ was active. Since at the limit, both tasks are considered, there is a discontinuity between the point $\alpha \to 0$ and the point $\alpha \ll \delta$.

B. Linear interpolation

This method consists in realizing a linear interpolation of the two control laws $\dot{\mathbf{q}}_{[\mathbf{1}|\mathbf{2}]}$ and $\dot{\mathbf{q}}_{[\mathbf{2}|\mathbf{1}]}$.

$$\dot{\mathbf{q}} = \alpha\dot{\mathbf{q}}_{[\mathbf{2}|\mathbf{1}]} + (1-\alpha)\dot{\mathbf{q}}_{[\mathbf{1}|\mathbf{2}]} \quad (18)$$

where α is the swap function, which is a smooth function of time. Using the damped definition of the control law, it is obvious that the continuity is ensured, even in the singular cases. Yet, this method presents two flaws.

First, since it is a blending of two control laws, this method does not write as a minimization problem. Hence, there is no guarantee that the robot motion is doable. Second, the computation time is increased, since the control has to be computed twice for these two levels of the stack of tasks. Namely, two extra computation of the inverse $(\mathbf{JP})^\dagger$ are required. Note that even if the stack contains n tasks, only these two levels must be computed twice, and not the entire stack. Despite these two flaws, this method has the advantage to be robust to any discontinuity and is easy to implement.

C. Extension to n tasks

Until now, the swap was realized between two consecutive tasks $\mathbf{e_1}$ and $\mathbf{e_2}$. This can be generalized to a swap between any consecutive tasks of the stack $\mathbf{e_i}$ and $\mathbf{e_{i+1}}$. Indeed, only the method to compute the control of tasks $\mathbf{e_i}$ and $\mathbf{e_{i+1}}$ changed. The tasks of higher priority $\mathbf{e_1}, \ldots, \mathbf{e_{i-1}}$ are not affected (by definition), and the lower priority tasks $\mathbf{e_{i+2}}, \ldots, \mathbf{e_n}$ still obey (5) due to the invariance of their null space associated: neither $S_{i+2} = S_i \cap \text{null}(\mathbf{J_i}) \cap \text{null}(\mathbf{J_{i+1}})$ nor $S_{i+3}, \ldots, S_n$ are modified (by recursion).

In order to ensure the continuity of the control, the swap of two tasks of arbitrary priority $\mathbf{e_i}$ and $\mathbf{e_{j,j>i}}$ is realized as follows. If all tasks $\mathbf{e_i}, \ldots, \mathbf{e_j}$ are compatible in the null space S_i, the priority of the tasks has no influence on the control value, so this operation can be realized instantly while maintaining a continuous control. In the other case, it is necessary to realize successive swaps between priority-nearby tasks in order to respect the tasks hierarchy: else way, the blending is likely to cause a modification of the behavior of the tasks in-between, which is not wanted.

V. Activation and deactivation of tasks

As mentioned in Section III, a hierarchy of tasks is ended by a virtual task that ensures the least-square norm and occupies all the null space left by other tasks. Since the null space left is empty, a task placed after in the priority order has no effect on the control. A task activation can then be performed by inserting the new task after the least-square task and exchanging their priority, and a task deactivation corresponds to the symmetric operation. Both operations can be done smoothly using the *smooth swap*.

This least-square task can be expressed explicitly under a minimization formulation. However it is implicit when using the pseudo-inverse approach. When swapping its priority with a task $\mathbf{e_{n+1}}$ to be added or removed, Eq. (18) is simply rewritten as a weight on the additional term corresponding to the inserted/removed task, with α the smooth factor varying from 0 to 1 (activation) or from 1 to 0 (removal):

$$\dot{\mathbf{q}}_\mathbf{n+1} = \dot{\mathbf{q}}_\mathbf{n} + \alpha(\mathbf{J_{n+1}}\mathbf{P_n})^+(\dot{\mathbf{e}}_\mathbf{n+1}^* - \mathbf{J_{n+1}}\dot{\mathbf{q}}_\mathbf{n}) \quad (19)$$

VI. Experiments

The smooth sequencing is exemplified by performing a task sequence with the HRP-2 humanoid robot. First, we show in simulation the differences of the trajectories generated with or without the smoothing.

A. Simulation

In this sequence, the robot has to realize three position tasks of the head ($\mathbf{e_H}$), the right arm ($\mathbf{e_R}$) and the left arm ($\mathbf{e_L}$), while keeping the waist at a fixed position. For each task, the reference is given as a position and rotation to reach. Each task is thus using 6 DOF of the robot, yet only 4 DOF are available for the task $\mathbf{e_H}$. These tasks are thus not compatible: in practice, they share the joints in the robot chest (2 DOF). The tasks are inserted and removed by order of insertion: we introduce the task $\mathbf{e_H}$ first, then $\mathbf{e_R}$ and finally $\mathbf{e_L}$, then remove the tasks one by one in the same order. The interval of presence inside the stack of each task is: $t_{\mathbf{e_H}} = [0.5, 2]$s, $t_{\mathbf{e_R}} = [1, 3]$s and $t_{\mathbf{e_L}} = [1.5, 3.5]$s.

The Fig. 2 shows the evolution of the non-smooth control law. Each insertion causes a discontinuity in the control law, that is due to the new contribution that each new task adds as an extra control value. Similarly, each removal produces another discontinuity. As explained in the upper sections, removing the task $\mathbf{e_H}$ causes in fact a double discontinuity, due to the removal of the task own contribution, and simultaneously to the change of the rank of the projected Jacobian of the tasks of less priority, that spares some DOF with the disappearance of $\mathbf{e_H}$. This double discontinuity can be noticed in the evolution of the trajectory of the velocity in

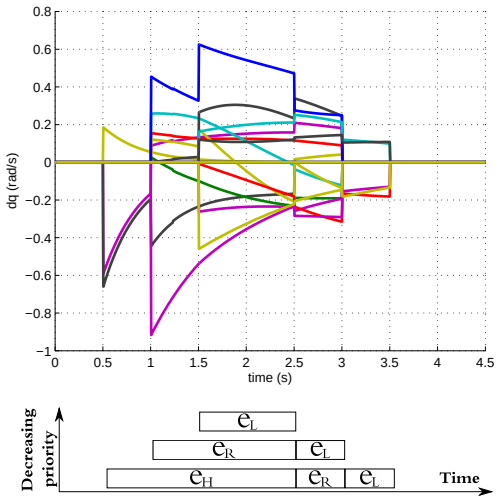

Fig. 2. Discontinuous evolution of the control law when using the classic stack-of-tasks computation. For each task add and removal, a discontinuity of the global control law can be noticed. The lower figure shows the content of the stack. The lower the task, the higher the priority.

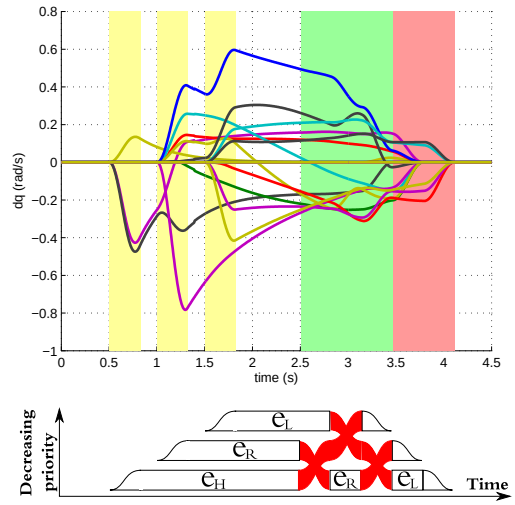

Fig. 4. Continuous evolution of the control law obtained by smoothing the realization of discrete events. Yellow (resp. red) zones indicate a smoothing during the insertion (resp. removal) of the last task of the stack. Green areas represent a smooth swap of priority between neighboring tasks. In the lower graph, red crosses represent the swap of two tasks.

the task space of $\mathbf{e_L}$, in Fig. 3: before time 2.5s, the task is in singularities due to $\mathbf{e_H}$, and the singularity suddenly disappears at $t = 2.5$s. When removing $\mathbf{e_R}$, the discontinuity is *only* due to the disappearance of the task contribution, since $\mathbf{e_L}$ and $\mathbf{e_R}$ are not incompatible when $\mathbf{e_H}$ is not active.

The Fig. 4 shows the smooth evolution of the control law, and the smoothing operations realized. The duration of the smoothing process is determined by the user, and is fixed to 0.3s in this experiment. As explained upper, only the least-priority task can be removed. A task must then first be swapped with lower-priority tasks. Task $\mathbf{e_H}$ starts the removal operation at $t = 2.5$s, swaps with $\mathbf{e_R}$ until $t = 2.8$s, then swaps with $\mathbf{e_L}$ until $t = 3.1$s and is finally removed since it has the least priority. These swaps avoid discontinuities in both the control law and the evolution in the $\mathbf{e_L}$ task space, as can be seen in Fig. 5. During the removal phase of $\mathbf{e_H}$, the task $\mathbf{e_R}$ starts its removal operation at $t = 3$s, swaps with $\mathbf{e_L}$ and is finally removed. Only one swap operation was necessary. Finally, $\mathbf{e_L}$ can be directly smoothly removed, being the last task in the stack.

When $\mathbf{e_R}$ is removed, it is not incompatible with $\mathbf{e_L}$. The swap period is then not necessary, since the order between two compatible tasks is not significant. These swap phases could be easily detected as unnecessary by comparing the two terms of (18), reducing thus the time necessary to effectively remove the stack of the task. Similarly, swapping $\mathbf{e_H}$ and $\mathbf{e_R}$ at $t = 2.5$s would have been avoided.

This experience emphasizes the necessity of a delay for removing a task: when a task add/removal causes a double discontinuity by both its control-law contribution and in the projection rank of lower-priority tasks, the double of time is needed to ensure the continuity.

B. Application to a real robot

These two motions are realized on the real HRP-2 robot. It is a 32-DOF humanoid, with position-based proportional controllers running at 500Hz. The inner control loop uses high-gain PD position control, which accounts for dynamical effects (gravity, friction, inertia, etc).

Our stack-of-task controller outputs a reference velocity. The velocity is simply integrated to obtain a reference position given to the robot low-level control loop. The two control laws (discontinuous and continuous) are applied on the robot, starting from the same initial position, displayed in Fig. 6. Since a discontinuity of the velocity would require an impossible infinite acceleration, but is absorbed by the

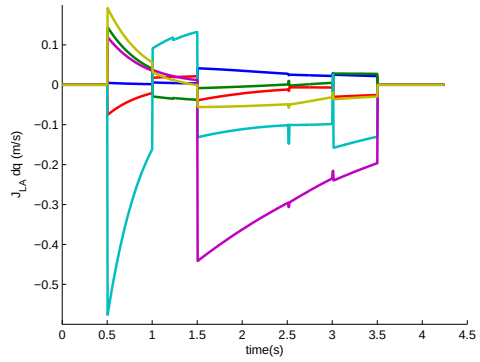

Fig. 3. Discontinuity of the velocity in task space $\mathbf{e_L}$. Before t=2.5s, $\mathbf{e_L}$ is in singularities due to upper-priority task. The singularity suddenly disappears when $\mathbf{e_H}$ is removed, causing a discontinuity.

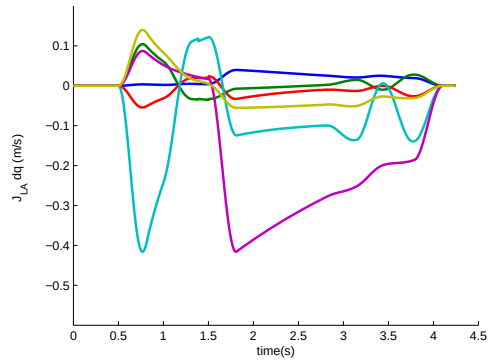

Fig. 5. The velocity in the task space of $\mathbf{e_L}$ is the projection of the continuous control law. It is thus continuous, even when removing the task $\mathbf{e_H}$ at time $t = 2.5$s.

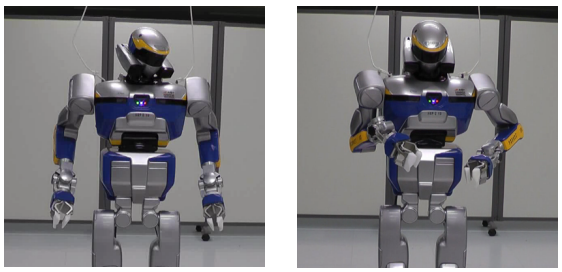

Fig. 6. Initial and final states of the robot.

robot mechanical, actuators physics, etc. The purpose of these experiments is to evaluate if the robot tracks better a desired posture when it corresponds to a smooth control.

We focus on one joint of the robot: the chest joint at the insertion of the first task. The Fig. 7 represents the evolution of the reference velocity in the classic and smooth cases. When following the motion, the position of the joint encoders are recorded. Fig. 8 shows the error observed between the reference position and obtained position. When using a continuous control law, the tracking is not perfect. This is due to the fact that it is not possible on this robot to provide the reference velocity to the low-level controller. A tracking error then appears while the velocity is important, and is reduced to 0 when the robot stops. Similarly, a residual error appears due to the classical absence of integral terms on HRP-2. With respect to the absence of proportional and derivative terms, the result of the tracking is very good. On the contrary, the tracking of the discontinuous control law is very poor: the discontinuity causes the excitation of vibrations, that are stabilized after some seconds.

VII. Conclusion

Classical hierarchy-of-task control formalisms ensure a continuous control law, even when crossing a singularity. This is not the case when the stack-of-task controller operate under on-the-fly discreet scheduling operation such as: adding or removing a task, or swapping the priority order between tasks. Subsequently instant task scheduling cannot be realized and a smoothing process is required. In this work, we first outlined this problem, and then proposed a generic solution, using a the swap between priority-nearby tasks as the basic operation than can be composed to produce an add or a removal of tasks at any priority. The obtained continuity can be tuned by increasing the duration of the smoothing period, as a trade-off between reactivity and smoothness. The proposed solution is generic, and can be applied for any set of tasks, or any pseudo-inverse based control schemes. In particular, it would be interesting to validate it for dynamic-inverse control scheme.

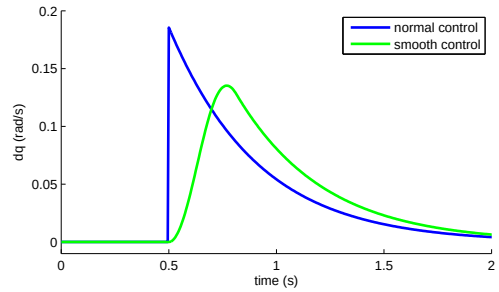

Fig. 7. Detail of the evolution of the control law for the chest joint using the smooth control law and the classic one.

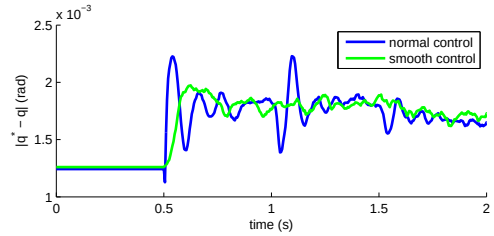

Fig. 8. Evolution of the difference between the desired posture q^* and the actual posture q for the chest joint. The task is inserted at t=0.5s

References

[1] P. Baerlocher. *Inverse kinematics techniques for the interactive posture control of articulated figures*. PhD thesis, EPFL, 2001.

[2] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. Robot. Autom.*, 13(3):398–410, June 1997.

[3] W. Decré, R. Smits, H. Bruyninckx, and J. De Schutter. Extending iTaSC to support inequality constraints and non-instantaneous task specification. In *IEEE Int. Conf. Robot. Autom. (ICRA'09)*, pages 1875–1882, Piscataway, NJ, USA, 2009. IEEE Press.

[4] K. Doty, C. Melchiorri, and C. Bonivento. A theory of generalized inverses applied to robotics. *Int. Jour. of Robotics Research (IJRR)*, 12(1):1–19, Dec. 1993.

[5] A. Escande, N. Mansard, and P.-B. Wieber. Fast resolution of hierarchized inverse kinematics with inequality constraints. In *IEEE Int. Conf. Robot. Autom. (ICRA'10)*, pages 3733–3738, Anchorage, USA, May 2010.

[6] O. Kanoun, F. Lamiraux, P.-B. Wieber, F. Kanehiro, E. Yoshida, and J.-P. Laumond. Prioritizing linear equality and inequality systems: application to local motion planning for redundant robots. In *IEEE Int. Conf. Robot. Autom. (ICRA'09)*, pages 724–729, Piscataway, NJ, USA, 2009. IEEE Press.

[7] F. Keith, N. Mansard, S. Miossec, and A. Kheddar. Optimization of tasks warping and scheduling for smooth sequencing of robotic actions. In *IEEE/RSJ Int. Conf. Intelligent Rob. Sys. (IROS'09)*, pages 1609–1614, october 2009.

[8] O. Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Int. Jour. of Robotics Research (IJRR)*, 3(1):43–53, Feb. 1987.

[9] J. Lee, N. Mansard, and J. Park. Intermediate desired value approach for continuous transition among multiple tasks of robots. In *IEEE Int. Conf. Robot. Autom. (ICRA'11)*, Shangai, China, May 2011.

[10] N. Mansard and F. Chaumette. Task sequencing for sensor-based control. *IEEE Trans. on Robotics*, 23(1):60–72, Feb. 2007.

[11] N. Mansard, A. Remazeilles, and F. Chaumette. Continuity of varying-feature-set control laws. *IEEE Trans. on Automatic Control*, 54(11):2493 – 2505, November 2009.

[12] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar. A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks. *Int. Conf. on Adv. Rob. (ICAR)*, 14(119):1–6, June 2009.

[13] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *Int. Jour. of Robotics Research (IJRR)*, 6(2):3–15, Feb. 1987.

[14] J. Salini, S. Barthélemy, and P. Bidaud. LQP-based controller design for humanoid whole-body motion. *Adv. in Rob. Kin.*, 3:177–184, 2010.

[15] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*. Clarendon Press, Oxford, United Kingdom, 1991.

[16] L. Sentis and O. Khatib. A whole-body control framework for humanoids operating in human environments. In *IEEE Int. Conf. Robot. Autom. (ICRA'06)*, pages 2641–2648, Orlando, USA, May 2006.

[17] B. Siciliano and J-J. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Int. Conf. on Adv. Rob. (ICAR)*, volume 2, pages 1211–1216, Pisa, Italy, June 1991.

[18] C. Van Loan. On the method of weighting for equality constrained least squares problems. Technical report, Cornell University, Ithaca, NY, USA, march 1984.