Prioritized Motion-Force Control of Constrained Fully-Actuated Robots: "Task Space Inverse Dynamics"

Andrea Del Prete[1,2], Francesco Nori[2], Giorgio Metta[2] and Lorenzo Natale[2]

1 — CNRS, LAAS, 7 avenue du colonel Roche, Univ de Toulouse, F-31400 Toulouse, France

2 — Istituto Italiano di Tecnologia, Via Morego 30, Genova, Italy

Abstract

We present a new framework for prioritized multi-task motion/force control of fully-actuated robots. This work is established on a careful review and comparison of the state of the art. Some control frameworks are not optimal, that is they do not find the optimal solution for the secondary tasks. Other frameworks are optimal, but they tackle the control problem at kinematic level, hence they neglect the robot dynamics and they do not allow for force control. Still other frameworks are optimal and consider force control, but they are computationally less efficient than ours. Our final claim is that, for fully-actuated robots, computing the operational-space inverse dynamics is equivalent to computing the inverse kinematics (at acceleration level) and then the joint-space inverse dynamics. Thanks to this fact, our control framework can efficiently compute the optimal solution by decoupling kinematics and dynamics of the robot. We take into account: motion and force control, soft and rigid contacts, free and constrained robots. Tests in simulation validate our control framework, comparing it with other state-of-the-art equivalent frameworks and showing remarkable improvements in optimality and efficiency.

Keywords: prioritized control, hierarchical control, inverse dynamics, force control, operational space, task space

1. Introduction

Several frameworks for multi-task control of rigid robots exist in the literature. Most frameworks presented in the '80s and '90s [1, 2, 3, 4] work at kinematic level, computing the desired joint velocities or accelerations. These approaches are not suited to robots that interact with the environment, because they do not allow for force control or impedance control. This motivated a more recent trend of torque control strategies [5, 6, 7, 8], which consider the dynamics of the robot, computing the desired joint torques. This approach can also improve tracking, because it compensates for the dynamic coupling between the joints of the multi-body system.

Peters et al. [9] showed that we can derive several of these well-known torque control laws under a Unifying Framework (UF), as solutions of constrained minimization problems. However, it is still unclear how these frameworks differ from each other and what their pros and cons are. This paper has a twofold aim: first, to provide a fair comparison of the state-of-the-art torque control frameworks; second, to present a new framework which outperforms the current state of the art. Our evaluation is based on four criteria: soundness, optimality, capabilities and efficiency. We carry out an analytical analysis of the frameworks and we test them in simulation to confirm the theoretical results.

Section 2 defines the basic tracking control problem and presents the main contribution of the paper through a simple example. Section 3 summarizes the related works and defines the notion of soundness, optimality, capabilities and efficiency. Section 4 and 5 describe the Unifying Framework (UF) [9] and the Whole-Body Control Framework (WBCF) [5], which are the frameworks that we chose for comparison, because they well represent the state of the art. Section 6 motivates the need for our new control framework Task Space Inverse Dynamics (TSID), which we then present in Section 7. For each framework we first discuss the solution for a single motion-control task, then we extend it to the multi-task case, and finally we introduce force control. Section 8 tests the three frameworks (TSID, UF, WBCF) in simulation on the same multi-task scenario, comparing their performances in terms of optimality and efficiency. The results prove that our control framework is sound, optimal and computationally more efficient than the other frameworks.

2. Key Idea

2.1. Notation and Problem Definition

We indicate with $\mathbb{S}_+^n$ the set of symmetric positive-definite $n \times n$ matrices. We want to design position tracking control laws for a rigid manipulator with n Degrees of Freedom (DoFs). The equation of motion of a manipulator in free space may be written as [10]:

$$M(q)\ddot{q} + h(q, \dot{q}) = \tau, \tag{1}$$

where $q \in \mathbb{R}^n$ are the generalized coordinates (e.g. joint angles), $\tau \in \mathbb{R}^n$ are the generalized forces (e.g. joint torques), $M(q) \in \mathbb{S}_+^n$ is the joint-space mass matrix, and $h(q, \dot{q}) \in \mathbb{R}^n$ contains all the nonlinear terms such as Coriolis, centrifugal and gravity forces. A position tracking task for the robot is described as a time-varying constraint $f(q) = x_r(t)$, where $x_r(t) \in \mathbb{R}^m$ is the reference task trajectory and $f : \mathbb{R}^n \to \mathbb{R}^m$ is a generic function of the generalized coordinates (e.g. the forward kinematics). Since we assume that the control inputs are the generalized forces τ, instantaneously we can only affect the generalized accelerations $\ddot{q}$. To express the task in terms of $\ddot{q}$ we differentiate the constraint twice with respect to time:

$$J(q)\dot{q} = \dot{x}_r(t), \qquad J(q)\ddot{q} + \dot{J}(q)\dot{q} = \ddot{x}_r(t),$$

where $J(q) = \frac{\partial}{\partial q} f(q) \in \mathbb{R}^{m \times n}$ is the task Jacobian. In the following, dependency upon t, q and $\dot{q}$ is no longer denoted to simplify notation. Since we use the second derivative of the constraint, in real situations a drift is likely to occur. To prevent deviations from the desired trajectory and to ensure disturbance rejection, we design a proportional-derivative feedback control law:

$$\ddot{x}^* = \ddot{x}_r + K_d(\dot{x}_r - \dot{x}) + K_p(x_r - x),$$

where $\ddot{x}^* \in \mathbb{R}^m$ is the desired task acceleration, whereas $K_d \in \mathbb{S}_+^m$ and $K_p \in \mathbb{S}_+^m$ are the derivative and proportional gain matrices, respectively.

2.2. A Simple Example

Following the approach taken in Peters et al. [9], we can derive task-space control laws as solutions of a constrained Quadratic Program (QP). To better convey the idea, in this example we take a simplified form of the dynamics and kinematics:

$$\tau^* = \underset{\tau \in \mathbb{R}^n}{\operatorname{argmin}} \, ||\ddot{x} - \ddot{x}^*||^2 \quad \text{s. t.} \qquad M\ddot{q} = \tau, \qquad J\ddot{q} = \ddot{x}$$

By resolving the constraints we can transform this QP into:

$$\tau^* = \underset{\tau \in \mathbb{R}^n}{\operatorname{argmin}} ||JM^{-1}\tau - \ddot{x}^*||^2$$

To solve this QP we can use the pseudoinverse [1] (for the sake of simplicity here we neglect null-space terms):

$$\tau^* = (JM^{-1})^{\dagger}\ddot{x}^* \tag{2}$$

Alternatively we can use a weighted pseudoinverse [11]:

$$\tau^* = (JM^{-1})^{\dagger_W}\ddot{x}^* = WM^{-1}J^T(JM^{-1}WM^{-1}J^T)^{\dagger}\ddot{x}^*,$$

where $W \in \mathbb{S}^n_+$ is an arbitrary weight matrix. The key idea that we are going to exploit is that a careful choice of W can lead to a more efficient solution. In particular, if we set $W = M^2$ we get:

$$\tau^* = MJ^T(JJ^T)^{\dagger}\ddot{x}^* = MJ^{\dagger}\ddot{x}^*$$

This expression has a lower computational cost than (2), mainly because it does not contain the inverse of the mass matrix M. In general the choice of W affects which value we select among the infinite solutions [11], so one could argue that a particular W leads to a solution that is somehow *better* than others [12]. However, in case of multi-task control we do not solve only one QP, but a sequence of QPs, which must have a *unique solution* (see Section 6.1 for details). In this case W does not affect which solution we select (because there is only one), so we claim that the choice of W should be aimed only at simplifying the computational cost of the solution.

3. Related Works

This section provides an overview of the related works. Table 1 lists the main features of the control frameworks that we considered in our analysis. We assess a control framework in terms of soundness, optimality, capabilities and efficiency. Table 1 also specifies the motor commands computed by each framework (column "Output"), which can be either joint torques τ, velocities $\dot{q}$ or accelerations $\ddot{q}$. A framework is *sound* if the control action of any task does not affect the performance of any higher priority tasks. A framework is *optimal* if its control action minimizes the error of each

Table 1: Control frameworks.

Framework	Optimal	Efficient	Force Control	Under actuated	Inequality	Output
Task Space Inverse Dynamics (TSID)	×	×	×			τ
Peters et al. [9] (UF)		×	×			τ
Sentis and Khatib [5] (WBCF)	×		×	(×)		τ
Mistry and Righetti [8]			×	×		τ
Saab et al. [7](SoT)	×		×	×	×	τ
De Lasa and Hertzmann [6]	×		×	×		τ
Jeong [13]	×	×				$\tau/\ddot{q}$
Smits et al. [14] (iTaSC)	×	×			×	$\dot{q}$
Chiaverini [3]		×				$\dot{q}$
Siciliano and Slotine [2]	×	×				$\dot{q}$
Baerlocher and Boulic [4]	×	×				$\dot{q}$
Nakamura et al. [1]		×				$\dot{q}/\ddot{q}$

task, under the constraint of being *sound*. The *capabilities* of a framework concern the types of tasks and systems that it allows to control. Finally, a framework is *efficient* if its computational complexity is minimal, considered its capabilities (typically, the more capabilities, the higher the computational complexity). In this context, *efficiency* is strictly related to the number of computed (pseudo)inverses, matrix multiplications and the computation of the mass matrix M.

All the control frameworks that we analyzed are *sound*. In terms of *capabilities*, Table 1 reports whether a framework allows for force control, whether it can control underactuated systems and whether it handles inequality constraints. Since we are interested in controlling robots that interact with the environment we focus on frameworks that allow for force control. Inequalities allow the user to define a task in terms of upper/lower bounds (e.g. joint limit avoidance, balance, visibility, and collision avoidance). However, this feature comes at a price: the algorithm can no longer compute the solution using pseudoinverses, but it requires a QP solver. Escande et al. [15] reached a computation time of 1 ms on an inverse-kinematics problem — at the price

of seldom suboptimal solutions. However, they did not consider the inverse-dynamics problem (as we do in this work), which has more than twice the number of variables and, consequently, is more computationally demanding. In another recent work Herzog et al. [16] succeeded in controlling their robot at 1 KHz using an inverse-dynamics formulation. Nonetheless they used a fast CPU (3.4 GHz) and the robot had only 14 DoFs; in case of more DoFs or slower CPU their method may still be too slow. For these reasons our control framework does not include inequality constraints, even if we believe that our results could be easily generalized to handle inequalities.

The possibility to control systems that are underactuated is crucial for real-world applications, however, for the sake of conciseness, this work deals only with fully-actuated systems. Another paper will present our results for underactuated robots. Besides the works cited in Table 1, another interesting approach for underactuated robots is presented in [17]. The authors select the contact forces based on the desired rate of change of the centroidal momentum, then they find desired joint accelerations that are consistent with these contact forces and finally they compute the joint torques using a hybrid-dynamics algorithm. Stephens and Atkeson [18] took a similar approach, with the main difference that they found joint accelerations and torques at the same time, resulting in a less efficient computation.

This work is motivated by the fact that no control framework that allows for force control is both optimal and efficient. Between the five frameworks that allow for force control, we select two as representative of the state of the art and we describe them in the next sections. Our first choice is the Unifying Framework (UF) [9], because it is the only one that allows for force control while being *efficient*. Our second choice is the Whole-Body Control Framework (WBCF) [5], because it represents the category of "optimal but not efficient" frameworks (i.e. the frameworks [7] and [6]). Even though the WBCF was extended to floating-base systems, here we consider the formulation for fully-actuated robots presented in [5] and implemented in [19].

4. Unifying Framework (UF)

The Unifying Framework (UF) [9] formulates the control problem as a constrained minimization:

$$\tau^* = \underset{\tau \in \mathbb{R}^n}{\operatorname{argmin}} \, ||\ddot{x} - \ddot{x}^*||^2 \quad \text{s. t.} \qquad \begin{aligned} M\ddot{q} + h &= \tau \\ J\ddot{q} + \dot{J}\dot{q} &= \ddot{x} \end{aligned} \tag{3}$$

For the typical case $m < n$, this problem has infinite solutions [11]:

$$\tau^* = (JM^{-1})^{\dagger_V}(\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h) + (I - (JM^{-1})^{\dagger_V}JM^{-1})\tau_0, \qquad (4)$$

where $\tau_0 \in \mathbb{R}^n$ is an arbitrary vector, $V \in \mathbb{S}^n_+$ is an arbitrary matrix and $A^{\dagger_V} = V^{\frac{1}{2}}(AV^{\frac{1}{2}})^\dagger = VA^T(AVA^T)^\dagger$ is the pseudoinverse of the matrix A, weighted by V. If A is full rank, then we can also write $A^{\dagger_V} = VA^T(AVA^T)^{-1}$. Choosing a particular pair (V, τ_0) we get the solution that minimizes $||V^{-\frac{1}{2}}(\tau - \tau_0)||^2$ [20]. Setting $\tau_0 = 0$ and varying V, we get different well-known control laws, reported in Table 2; the second row reports the Operational Space control

Table 2: Control laws for different values of weight matrix V

V	minimize	Control law, τ^*
I	$\|\|\tau\|\|^2$	$M^{-1}J^T(JM^{-2}J^T)^\dagger(\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h)$
M	$\tau^T M^{-1}\tau$	$J^T(JM^{-1}J^T)^\dagger(\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h)$
M^2	$\|\|M^{-1}\tau\|\|^2$	$MJ^T(JJ^T)^\dagger(\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h)$

law of Khatib [21], which selects the torques that could be generated by a hypothetical force applied at the control point. Without loss of generality, given that $M \in \mathbb{S}^n_+$, we can set $V = M^2W$, where $W \in \mathbb{S}^n_+$ is another arbitrary matrix, so that (4) simplifies to:

$$\tau^* = MJ^{\dagger_W}(\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h) + MN^W M^{-1}\tau_0, \qquad (5)$$

where $N^W = I - J^{\dagger_W}J$ is a weighted (nonorthogonal) null-space projector.

4.1. Hierarchical Extension

The Unifying Framework can manage an arbitrary number of tasks N, each characterized by a desired acceleration $\ddot{x}^*_i$ and a Jacobian J_i. To ensure the correct management of task conflicts, the tasks need prioritization: the higher the number i of the task, the higher its priority.

$$\begin{aligned}
\tau^* &= M\ddot{q}_1 \\
\ddot{q}_i &= \ddot{q}_{i+1} + N^W_{p(i)}J_i^{\dagger_W}(\ddot{x}^*_i - \dot{J}_i\dot{q} + J_iM^{-1}h) \qquad i \in [1, N] \\
N^W_{p(i)} &= N^W_{p(i+1)} - (J_{i+1}N^W_{p(i+1)})^{\dagger_W}J_{i+1}N^W_{p(i+1)},
\end{aligned}$$

where $N^W_{p(i)}$ is a projector into the null space of all the tasks $\{j \,|\, j > i\}$, computed with the recursive formula proposed in [4]. The algorithm starts by computing $\ddot{q}_N$ (using $\ddot{q}_{N+1} = 0$ and $N^W_{p(N)} = I$) and it proceeds backwards up to $\ddot{q}_1$. If the state of the robot is completely controlled, which is usually the case (see Section 6.1), then this formulation simplifies to:

$$\begin{aligned} \tau^* &= M\ddot{q}_1 + h \\ \ddot{q}_i &= \ddot{q}_{i+1} + N^W_{p(i)} J_i^{\dagger W} (\ddot{x}^*_i - \dot{J}_i \dot{q}) \qquad i \in [1, N] \end{aligned} \tag{6}$$

The accelerations of each task $\ddot{q}_i$ are projected into the null space of the higher priority tasks; this guarantees that the framework is *sound*. However, this approach is not *optimal*, because each task is solved independently, and then projected into the null space of the higher priority tasks. This does not ensure the minimization of the error of each task (see [4, 3] for a thorough explanation).

4.2. Hybrid Control

The Unifying Framework allows for hybrid position/force control by setting the joint space control torques to:

$$\tau_0 = h - J_c^T f^*,$$

where $J_c(q) \in \mathbb{R}^{k \times n}$ is the contact Jacobian, $f^* \in \mathbb{R}^k$ are the desired contact forces and $k \in \mathbb{R}$ is the number of independent directions in which the robot applies force. Substituting τ_0 into the desired control torques (5) we get:

$$\tau^* = M J^{\dagger W} (\ddot{x}^* - \dot{J}\dot{q}) + h - M N^W M^{-1} J_c^T f^*,$$

where the applied forces act in the null space of the tracking task.

5. Whole-Body Control Framework (WBCF)

In this section we describe the WBCF presented by Sentis and Khatib [5]. This framework is based on the Operational Space Formulation [21], which we can derive by setting $W = M^{-1}$ in (5):

$$\tau^* = J^T \underbrace{(JM^{-1}J^T)^\dagger}_{\Lambda} (\ddot{x}^* - \dot{J}\dot{q} + JM^{-1}h) + (I - J^T J^{T\dagger_{M^{-1}}})\tau_0,$$

where $J^{\dagger_{M^{-1}}} = M^{-1}J^T\Lambda$ is the dynamically-consistent Jacobian pseudoinverse and Λ is the task-space mass matrix.

5.1. Hierarchical Extension

While in case of a single task the WBCF and the UF are equivalent, their hierarchical extensions differ substantially:

$$\begin{aligned} \tau^* &= \sum_{i=1}^{N} J_{p(i)}^T F_{p(i)} \\ F_{p(i)} &= \Lambda_{p(i)}(\ddot{x}_i^* - \dot{J}_i\dot{q} + J_i M^{-1}(h - \sum_{j=1}^{i-1} J_{p(j)}^T F_{p(j)})) \\ J_{p(i)} &= J_i(I - \sum_{j=1}^{i-1} J_{p(j)}^{\dagger_{M^{-1}}} J_{p(j)}) \end{aligned} \tag{7}$$

This prioritization strategy is different with respect to (6): the WBCF minimizes the error of each task under the constraint of not conflicting with any higher priority tasks, namely it is *optimal*. However, this formulation is computationally less efficient than (6) because i) it contains the term M^{-1} and ii) it requires more matrix multiplications.

5.2. Hybrid Control

The WBCF allows for hybrid position/force control by setting:

$$F_{p(i)} = \Omega_f f_i^* + \Lambda_{p(i)}(\Omega_m \ddot{x}_i^* - \dot{J}_i\dot{q} + J_i M^{-1}(h - \sum_{j=1}^{i-1} J_{p(j)}^T F_{p(j)})),$$

where the selection matrices Ω_f and Ω_m split the control space into force and motion components, respectively.

6. The Need for a New Control Framework

The WBCF is *sound* and *optimal*, but it is not *efficient* because it requires the computation of the *operational space inertia matrices* Λ's. The simplest way to compute them is using the formula $\Lambda = (JM^{-1}J^T)^\dagger$, which has a complexity of $O(n^3)$: the computation of M — with Recursive Newton-Euler Algorithm (RNEA) or Composite-Rigid-Body algorithm [10] — has a complexity of $O(n^2)$ for serial robots and $O(nd)$ for multi-branch robots (where d is the tree depth). More efficient algorithms [22] can compute Λ with a complexity of $O(nm^2 + m^3)$, where m is the dimension of the task.

On the other hand the UF is *sound* and *efficient* (if we choose $V = M^2$, i.e. $W = I$): the solution takes the form $\tau^* = M\ddot{q}_1 + h$, which we can calculate without explicitly computing M, through the $\mathrm{O}(n)$ RNEA. Nonetheless, the UF is not *optimal*: even if a task does not conflict with any higher priority tasks, it may not be performed correctly.

The derivation of our framework, TSID, follows the same principles underlying the UF, but with a different hierarchical extension. We minimize the error of each task under the constraint of not affecting any higher priority task. At each minimization step, we carefully select the weight matrices used in the pseudoinverses, so as to simplify the resulting control laws. This leads to an efficient formulation, while preserving the optimality property. We start considering position tracking control only, then we introduce force control tasks.

6.1. Weight Matrix and Joint Space Stabilization

The weight matrix V (or equivalently W) introduced in the resolution of (3) can play two different roles. In case there is no secondary task (i.e. $\tau_0 = 0$), V determines the quantity that we minimize (e.g. $||\tau||^2$, $||\ddot{q}||^2$, $||M^{-\frac{1}{2}}\tau||^2$). In case there is a secondary task, V specifies the metric that is used to measure the distance between τ and τ_0.

Using the null space of a task to minimize some measure of effort is appealing, mainly because it is rooted in the study of human motion [23]. This approach may be feasible in simulation, but unfortunately in reality it leads to singular configurations and hitting of joint limits [9]. The subspace of joint accelerations that does not affect the task is not controlled, so its behavior is determined by disturbances and errors in the model of the manipulator. Even in simulation, if the robot has nonzero joint velocities when the controller starts, failing to use a secondary task may result in joint space instability. The reason for this behavior is obvious: the effort of stabilizing in joint space is not task relevant and it would increase the cost [9].

Peters et al. [9] suggest to add a joint space motor command for stabilization. A common approach is to design the postural task to attract the robot towards a desired posture q_p. We compute the desired joint accelerations as $\ddot{q}_p^* = K_p(q_p - q) - K_d\dot{q}$, where $K_p \in \mathbb{S}_+^n$ and $K_d \in \mathbb{S}_+^n$ are the proportional and damping gain matrices. In the following we always include the postural task to minimize $||\ddot{q} - \ddot{q}_p^*||^2$, under the constraint of not affecting any other task. This ensures stabilization of the manipulator in joint space.

7. Original Contribution - Task Space Inverse Dynamics (TSID)

In this section we derive the TSID control framework, which is the main contribution of the paper. The TSID is *sound*, *optimal*, *efficient* — as confirmed by the simulation tests — and allows for both motion and force control.

7.1. Framework Derivation

Consider a general scenario in which the robot has to perform N position tracking tasks $T_1 \dots T_N$ and a postural task T_0 (with desired joint accelerations $\ddot{q}_p^*$) to stabilize any left redundancy. Taking inspiration from the UF and from [6] we formulate the control problem as a sequence of constrained minimization, starting from the highest-priority task N and moving down to the lowest-priority task 0 (i.e. the postural task):

$$\begin{array}{lllll} (T_N) & g_N^* = \min\limits_{\tau\in\mathbb{R}^n} g_N(\tau) & \text{s.t.} & M\ddot{q}+h=\tau & \\ (T_i) & g_i^* = \min\limits_{\tau\in\mathbb{R}^n} g_i(\tau) & \text{s.t.} & M\ddot{q}+h=\tau, & g_j(\tau)=g_j^* \quad \forall j>i \\ (T_0) & \tau^* = \operatorname{argmin}\limits_{\tau\in\mathbb{R}^n} ||\ddot{q}-\ddot{q}_p^*|| & \text{s.t.} & M\ddot{q}+h=\tau, & g_j(\tau)=g_j^* \quad \forall j>0, \end{array} \tag{8}$$

where $g_i(\tau) = ||J_i\ddot{q} + \dot{J}_i\dot{q} - \ddot{x}_i^*||^2$ is the cost associated to the task T_i. The solution of (8) is given by:

$$\begin{aligned} \tau^* =& M\ddot{q}_0 + h \\ \ddot{q}_i =& \ddot{q}_{i+1} + N_{p(i)}^W (J_i N_{p(i)}^W)^{\dagger_W} (\ddot{x}_i^* - \dot{J}_i\dot{q} - J_i\ddot{q}_{i+1}) \qquad i\in[0,N], \end{aligned} \tag{9}$$

where $J_0 = I$ and $\ddot{x}_0^* = \ddot{q}_p^*$. The computation is initialized setting $\ddot{q}_{N+1} = 0$ and $N_{p(N)} = I$. Once again, selecting the weight matrix W we can vary the form of the control law. Interestingly enough though, the solution τ^* is independent of W. This is because the only role of W is to weight the quantity that is minimized in the null space of all the tasks, but here the postural task ensures that there is no null space left (because any control action affects the postural task). It is then reasonable to choose W so as to simplify the computation. If we set $W = I$ then all the null-space projectors $N_{p(i)}$ become orthogonal, so they are equal to their pseudoinverses (i.e. $N_{p(i)}^\dagger = N_{p(i)}$). This simplifies the formulation (9) to:

$$\begin{aligned} \tau^* =& M(\ddot{q}_1 + N_{p(0)}\ddot{q}_p^*) + h \\ \ddot{q}_i =& \ddot{q}_{i+1} + (J_i N_{p(i)})^\dagger (\ddot{x}_i^* - \dot{J}_i\dot{q} - J_i\ddot{q}_{i+1}) \qquad i\in[1,N] \end{aligned} \tag{10}$$

In this form, kinematics and dynamics are completely decoupled: first we solve the multi-task prioritization at kinematic level computing $\ddot{q}_1$, then we compute the torques to get the desired joint accelerations. This formulation does not require the computation of a pseudoinverse for the postural task, because it exploits the property of orthogonal projectors of being equal to their pseudoinverses. Moreover, it can be efficiently computed with the RNEA, without explicitly calculating M.

7.2. Force Control

This subsection extends TSID to force control. If the manipulator is in contact with the environment, its equations of motion become:

$$M(q)\ddot{q} + h(q, \dot{q}) - J_c(q)^T f = \tau, \tag{11}$$

where $J_c(q) = \frac{\partial x_c}{\partial q} \in \mathbb{R}^{k \times n}$ is the contact Jacobian (or constraint Jacobian), $x_c \in \mathbb{R}^k$ is the robot contact point and $f \in \mathbb{R}^k$ are the contact forces (or constraint forces). To control the contact forces we need a model of the contact dynamics. The most common choices are the *linear-spring contact* model [24] and the *rigid contact* model. The first model assumes that the environment at the contact point behaves like a linear spring, i.e. $k_s(x_c - x_e) = f$, where k_s is the contact stiffness and $x_e \in \mathbb{R}^k$ is the environment contact point. Assuming k_s is known, force is a known function of position, so we can easily translate this kind of force control problems into position control problems.

More interesting is instead the *rigid contact* model, mainly because it introduces constraints into the problem formulation. When the manipulator is in rigid contact with the environment, its motion is subject to k nonlinear constraints. In general we can consider these constraints as nonlinear function of the generalized coordinates, their derivatives and time: $c(q, \dot{q}, t) = 0$ [1]. To include these constraints into the control problem we express them at acceleration level[2] as: $J_c(q)\ddot{q} = b(q, \dot{q}, t)$. We write then the problem as:

$$\begin{aligned} \tau^* = \underset{\tau \in \mathbb{R}^n}{\operatorname{argmin}} \, ||f - f^*||^2 \quad \text{s. t.} \qquad & M\ddot{q} + h - J_c^T f = \tau \\ & J_c\ddot{q} = b, \end{aligned} \tag{12}$$

[1] The constraints may be time-varying, hence we can model contacts with curved surfaces, as long as c is sufficiently smooth.

[2] In case of nonholonomic constraints we differentiate them once, whereas in case of holonomic constraints we differentiate them twice. For instance, in case of time-invariant rigid contacts we have: $b(q, \dot{q}) = -\dot{J}_c(q, \dot{q})\dot{q}$.

where $f^* \in \mathbb{R}^k$ are the desired contact forces. We can express the infinite solutions of the problem (12) as:

$$\tau^* = M(J_c^\dagger b + N_c \ddot{q}_0) + h - J_c^T f^*, \tag{13}$$

where $\ddot{q}_0 \in \mathbb{R}^n$ is an arbitrary vector. It is trivial to show that (13) is a solution of (12) because it respects the constraints and it results in the minimum cost (i.e. 0). This control law is one of our main contributions, because it allows to implement force control without computing M, while characterizing the redundancy of the task through $\ddot{q}_0$.

7.3. Integration of Force Control in Hierarchical Framework

We extend the multi-task formulation (10) to include force control tasks. The rigid force control task, if any, has to take the highest priority because it is a physical constraint that cannot be violated by definition. We assume that the robot has to perform $N-1$ position control tasks. On top of that there is a rigid force control task N (for the sake of simplicity, here we assume holonomic constraints, i.e. $b(q, \dot{q}, t) = -\dot{J}_c \dot{q}$), with reference force f^* and Jacobian $J_N = J_c$:

$$\begin{aligned} \tau^* =& M(\ddot{q}_1 + N_{p(0)} \ddot{q}_p^*) + h - J_c^T f^* \\ \ddot{q}_i =& \ddot{q}_{i+1} + (J_i N_{p(i)})^\dagger (\ddot{x}_i^* - \dot{J}_i \dot{q} - J_i \ddot{q}_{i+1}) \qquad i \in [1, N], \end{aligned} \tag{14}$$

where $\ddot{x}_N^* = \ddot{x}_c = 0$, $\ddot{q}_{N+1} = 0$, and $N_{p(N)} = I$. Even after the extension to force control, kinematics and dynamics are still decoupled, so the computational complexity has not increased and τ^* can be efficiently computed with the RNEA.

8. Tests

8.1. Simulation Environment

We tested our control framework — Task Space Inverse Dynamics — against the Unifying Framework (UF) [9] and the Whole-Body Control Framework (WBCF) [5], on a customized version of the Compliant huManoid (CoMan) simulator [25]. The robot has 23 DoFs: 4 in each arm, 3 in the torso and 6 in each leg. We adapted the simulator to make the robot rigid and fully-actuated (we fixed the robot base and we removed the joint passive compliance). Direct and inverse dynamics, both in simulation and control,

were efficiently computed using C language functions, generated with the Robotran [26] symbolic engine. Contact forces were simulated using linear spring-damper models (stiffness $2 \cdot 10^5 N/m$ and damping $10^3 Ns/m$, as proposed in [25]) with realistic friction. To integrate the equations of motion we used the Simulink variable step integrator *ode23t*, with relative and absolute tolerance of 10^{-3} and 10^{-6}, respectively. The tests were executed on a computer with a 2.83 GHz CPU and 4 GB of RAM. The computation times are computed as averages over the whole test (i.e. some thousands of executions).

8.2. Trajectory Generation

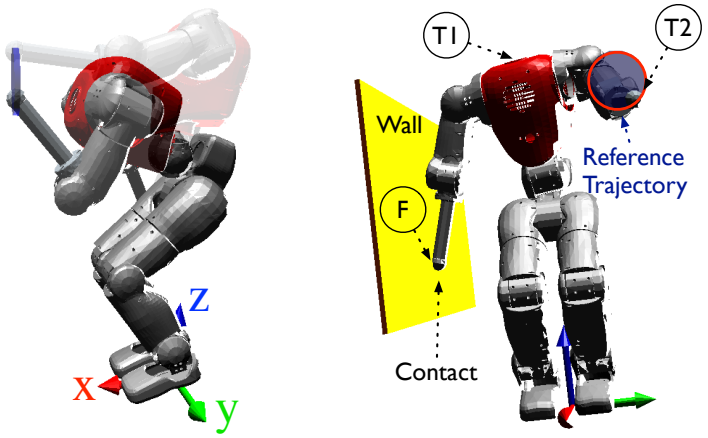
To generate reference position-velocity-acceleration trajectories we used the approach presented in [27], which provides approximately minimum-jerk trajectories. The trajectory generator is a 3rd order dynamical system that takes as input the desired trajectory $x_d(t)$ and outputs the three position-velocity-acceleration reference trajectories $x_r(t), \dot{x}_r(t), \ddot{x}_r(t)$. The reference position trajectory follows the desired position trajectory with a velocity that depends on the parameter "trajectory time" (always set to $1.0s$ in our tests). We set all proportional gains $K_p = 10s^{-2}$, and all derivative gains $K_d = 5s^{-1}$.

8.3. Damped Pseudoinverses

The controllers used damped pseudoinverses [3] to ensure stability near singularities. Based on our experience on a real robot, we set the damping factor $\lambda = 0.02$, which ensures a maximum gain of the pseudoinverses of $(2\lambda)^{-1} = 25$. To avoid interferences between tasks, null-space projection matrices were computed without any damping, but setting to zero all singular values below the threshold $\sigma_{min} = 2.5 \cdot 10^{-8}$. The values λ and σ_{min} must be chosen so that: $\sigma_{min}(\sigma_{min}^2 + \lambda^2)^{-1} < z$, where z is a small positive value (e.g. we chose $z \simeq 10^{-4}$). This ensures coherence between damped pseudoinverses and null-space projectors, so that any direction of the control space is not used by more tasks at the same time. The use of damped pseudoinverses modifies the minimization problem (8), adding a regularization term $\lambda||\tau||^2_W$ to the cost functions. Away from singularities this term has negligible effects, but close to singularities it keeps $||\tau^*||$ bounded, at the expenses of the task errors. If $\lambda = 0$, we know that WBCF and TSID give the same results. However, with $\lambda \neq 0$, since the regularization term is affected by the weight

matrix W, we expect to see some differences between WBCF and TSID when close to singularities.

8.4. Test 1 - Feasible Task Hierarchy



(a) (b)

Figure 1: CoMan [25] executing Test 1. Task F controls the force exerted by the right hand against the wall. Task T_2 moves the left hand along the circular reference trajectory depicted as a red circumference. Task T_1 moves the neck base back and forth along the x axis.

In this test the robot performs four tasks:

F**:** 3 DoFs, apply a normal force of 20 N on the wall with the right hand

T_2**:** 3 DoFs, track a circular trajectory with the left hand

T_1**:** 1 DoF (x coordinate), track a sinusoidal reference with the neck base

T_0**:** 23 DoFs, maintain the initial joint posture

The first three tasks are always compatible, so the robot should be able to perform them with negligible errors. Table 3 reports the root-mean-square error (RMSE) for each task and the mean computation time of the control loop. We compute the RMSE as $\sqrt{\frac{1}{N_t}\sum_{t=0}^{T}||x(t) - x_r(t)||^2}$, where N_t is the number of samples used in the summation. The criteria proposed in Section 3 (in particular see Table 1) are strictly connected to the data of Table 3:

the error of the primary task F concerns the *soundness*, the errors of the nonprimary tasks (T_2, T_1, T_0) concern the *optimality*, and the computation time concerns the *efficiency*. As expected, the UF performs poorly on the

Table 3: Test 1. Root-mean-square error of the four tasks and average computation time of the controller.

Related to	*Soundness*	*Optimality*			*Efficiency*
Controller	F-RMSE (N)	T_2-RMSE (mm)	T_1-RMSE (mm)	T_0-RMSE (°)	Computation Time (ms)
TSID	**0.1**	**0.4**	**0.1**	**7.1**	**0.24**
WBCF	0.1	0.4	0.1	7.1	0.64
UF	0.1	36.8	30.1	6.6	0.25

nonprimary tasks, because it is not optimal. Both WBCF and TSID achieve good tracking on all tasks, but the computation time of WBCF is $\sim 2.6\times$ the computation time of our framework.

8.5. Test 2 - Unfeasible Task Hierarchy

In this test the robot performs the same four tasks of the previous test, with the only difference that task T_1 controls the 3D Cartesian position of the neck base (rather than the x coordinate only). This makes impossible

Table 4: Test 2. Root-mean-square error of the four tasks and average computation time of the controller.

Related	*Soundness*	*Optimality*			*Efficiency*
Controller	F-RMSE (N)	T_2-RMSE (mm)	T_1-RMSE (mm)	T_0-RMSE (°)	Computation Time (ms)
TSID	**0.0**	**0.1**	**21.5**	**5.5**	**0.25**
WBCF	0.0	0.3	21.5	5.5	0.67
UF	0.0	23.8	62.4	5.1	0.26

to achieve all tasks at the same time (the desired neck trajectory is not reachable), so we expect a significant error for task T_1, while the tasks F and T_2 should have negligible errors. Table 4 shows that, as before, UF performs

poorly on the nonprimary tasks, whereas WBCF has higher computation time than the other two frameworks. The small difference between TSID and WBCF in the RMSE of task T_2 is due to the behavior of the damped pseudoinverses when close to the singularity due to the conflict between task T_1 and the tasks F and T_2. While this effect is clear from a numerical standpoint (see Section 8.3), there seems not to be a *best* choice for the weight to use for damping; indeed we observed that, when close to singularities, sometimes WBCF performs slightly worse than TSID, but other times it performs slightly better.

9. Conclusions

We presented and validated a new theoretical control framework, called Task Space Inverse Dynamics, for prioritized motion and force control of fully-actuated robots. To the best of our knowledge, this framework outperforms every other control framework with equal capabilities. Its main features are:

1. *optimality*: it minimizes the error of each task under the constraint of not affecting any higher priority task
2. *capabilities*: it allows for position/velocity/acceleration control and soft/rigid contact force control
3. *efficiency*: it computes the desired joint torques in $\mathrm{O}(n)$ using the Recursive Newton-Euler Algorithm because it needs neither the joint-space mass matrix M, nor the task-space mass matrices Λ's

We compared the presented control framework with other two state-of-the-art control frameworks (UF and WBCF), both analytically and through simulation tests. We decided to carry out this comparison in simulation to avoid that model inaccuracies could interfere with the controllers in unpredictable ways. The results confirm that our framework outperforms the other two frameworks, either in terms of optimality or efficiency.

Moreover this work proves that, for fully-actuated robots, it is not necessary to take into account the dynamics when resolving the multi-task motion/force control problem. In other words, the WBCF is equivalent to a second-order inverse kinematics — which computes the desired joint accelerations — followed by an inverse dynamics — which computes the desired joint torques.

9.1. Discussion and Future Work

The presented framework suffers from three main limitations, which we should address in practical applications.

1. It does not allow for inequality constraints [7, 14], which are particularly important for modeling joint limits and motor torque bounds.
2. It does not deal with planning and it guarantees only instantaneous (local) optimality, so a task may lead the robot into a configuration in which a higher priority task becomes singular, hence unfeasible. To tackle this issue the cost function should include not only instantaneous errors, but the summation of errors over a certain time horizon (i.e. model predictive control, MPC [28, 29]).
3. We only considered fully-actuated robots, while many mechanical systems are underactuated (e.g. floating-base, underwater, elastic robots).

While these limitations could make this contribution seem negligible, we argue the opposite. The frameworks that tackle the limitations 1 and 2 typically work by iteratively solving simplified control problems such as the ones we discussed (i.e. nonlinear functions are linearized around the current solution, whereas inequality constraints are converted into equality constraints using active-set methods). This implies that more advanced frameworks could exploit the presented results to improve their efficiency — which often is the bottleneck preventing their applications on real robots [29]. Regarding the limitation 3, for the sake of conciseness this work has dealt only with fully-actuated mechanical systems. A future paper will present an extension of this framework for floating-base robots, which relies on the same principles and techniques that we used here. Finally, we are now in the process of testing TSID on a real humanoid robot.

Acknowledgement

This paper was supported by the FP7 EU projects CoDyCo (No. 600716 ICT 2011.2.1 Cognitive Systems and Robotics), and Koroibot (No. 611909 ICT-2013.2.1 Cognitive Systems and Robotics).

References

[1] Y. Nakamura, H. Hanafusa, T. Yoshikawa, Task-Priority Based Redundancy Control of Robot Manipulators, The International Journal of Robotics Research 6 (1987) 3–15.

[2] B. Siciliano, J. J. E. Slotine, A general framework for managing multiple tasks in highly redundant robotic systems, in: Advanced Robotics, 'Robots in Unstructured Environments', 91 ICAR, Fifth International Conference on, IEEE, 1991, pp. 1211–1216.

[3] S. Chiaverini, Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators, IEEE Transactions on Robotics and Automation 13 (1997) 398–410.

[4] P. Baerlocher, R. Boulic, Task-priority formulations for the kinematic control of highly redundant articulated structures, Intelligent Robots and Systems (1998).

[5] L. Sentis, O. Khatib, Synthesis of whole-body behaviors through hierarchical control of behavioral primitives, International Journal of Humanoid Robotics 2 (2005) 505–518.

[6] M. De Lasa, A. Hertzmann, Prioritized optimization for task-space control, in: Intelligent Robots and Systems, IEEE/RSJ International Conference on, volume 3, Ieee, 2009, pp. 5755–5762.

[7] L. Saab, N. Mansard, F. Keith, J.-Y. Fourquet, P. Soueres, Generation of dynamic motion for anthropomorphic systems under prioritized equality and inequality constraints, Robotics and Automation, IEEE International Conference on (2011) 1091–1096.

[8] M. Mistry, L. Righetti, Operational Space Control of Constrained and Underactuated Systems, in: Proceedings of robotics: science and systems, 2011.

[9] J. Peters, M. Mistry, F. E. Udwadia, J. Nakanishi, S. Schaal, A unifying framework for robot control with redundant DOFs, Autonomous Robots 24 (2007) 1–12.

[10] B. Siciliano, O. Khatib, Springer Handbook of Robotics, volume 15, Springer, 2008.

[11] Y. Nakamura, Advanced robotics: redundancy and optimization, 1990.

[12] H. Bruyninckx, O. Khatib, Gauss' principle and the dynamics of redundant and constrained manipulators, Robotics and Automation (2000) 2563–2568.

[13] J. Jeong, A Task-priority Based Framework for Multiple Tasks in Highly Redundant Robots, Intelligent Robots and Systems, IEEE/RSJ International Conference on. (2009) 5886–5891.

[14] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, J. De Schutter, iTASC: a tool for multi-sensor integration in robot manipulation, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems 2 (2008) 426–433.

[15] A. Escande, N. Mansard, P.-B. Wieber, HQP, International Journal of Robotics Research (in press) (2014).

[16] A. Herzog, L. Righetti, F. Grimminger, Experiments with a hierarchical inverse dynamics controller on a torque-controlled humanoid, arXiv preprint arXiv:1305.2042 (2013).

[17] S.-H. Lee, A. Goswami, A momentum-based balance controller for humanoid robots on non-level and non-stationary ground, Autonomous Robots 33 (2012) 399–414.

[18] B. J. Stephens, C. G. Atkeson, Dynamic Balance Force Control for compliant humanoid robots, 2010.

[19] R. Philippsen, L. Sentis, O. Khatib, An open source extensible software package to create whole-body compliant skills in personal mobile manipulators, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, IEEE, 2011, pp. 1036–1041.

[20] A. Bjorck, Numerical methods for least squares problems, Society for Industrial Mathematics, 1996.

[21] O. Khatib, A unified approach for motion and force control of robot manipulators: The operational space formulation, IEEE Journal on Robotics and Automation 3 (1987) 43–53.

[22] K. Chang, O. Khatib, Operational space dynamics: efficient algorithms for modeling and control of branching mechanisms, in: Robotics and Automation. Proceedings. ICRA'00. IEEE International Conference on, volume 1, 2000, pp. 850–856.

[23] T. Flash, N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, The journal of Neuroscience 5 (1985) 1688–1703.

[24] J. Park, O. Khatib, Robot multiple contact control, Robotica 26 (2008) 667–677.

[25] H. Dallali, M. Mosadeghzad, G. A. Medrano-Cerda, N. Docquier, P. Kormushev, N. Tsagarakis, Z. Li, D. Caldwell, Development of a Dynamic Simulator for a Compliant Humanoid Robot Based on a Symbolic Multibody Approach, in: International Conference on Mechatronics, Vicenza, Italy, 2013.

[26] http://www.robotran.be (Robotran webpage), 2012.

[27] U. Pattacini, F. Nori, L. Natale, G. Metta, G. Sandini, An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, IEEE, 2010, pp. 1668–1674.

[28] I. R. Manchester, U. Mettin, F. Iida, R. Tedrake, Stable dynamic walking over uneven terrain, The International Journal of Robotics Research 30 (2011) 265–279.

[29] Y. Tassa, T. Erez, E. Todorov, Synthesis and stabilization of complex behaviors through online trajectory optimization, in: Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on, 2012, pp. 4906–4913.