Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

MPTC – Modular Passive Tracking Controller for stack of tasks based control frameworks

Johannes Englsberger, Alexander Dietrich, George Mesesan,
Gianluca Garofalo, Christian Ott and Alin Albu-Schäffer
Institute of Robotics and Mechatronics
German Aerospace Center (DLR), Oberpfaffenhofen, Germany
Email: johannes.englsberger@dlr.de

***Abstract*—This work introduces the so-called Modular Passive Tracking Controller (MPTC), a generic passivity-based controller, which aims at independently fulfilling several subtask objectives. These are combined in a stack of tasks (SoT) that serves as a basis for the synthesis of an overall system controller. The corresponding analysis and controller design are based on Lyapunov theory. An important contribution of this work is the design of a specific optimization weighting matrix that ensures passivity of an overdetermined and thus conflicting task setup. The proposed framework is validated through simulations and experiments for both fixed-base and free-floating robots.**

I. Introduction

Simultaneous control of multiple tasks has emerged as a major research topic in robotic control. While initial works considered the simpler case of a single task and its nullspace for a kinematically redundant robot, nowadays there exist several well established frameworks for handling multiple tasks with and without priorities. In the literature one may distinguish between works that solve the task coordination problem first on a *kinematic level*, and works that formulate the control *directly for the dynamics*. Another important classification can be done based on the use of *strict task priorities* via hierarchic controllers as compared to controllers which apply a *soft prioritization* via task weighting.

At the *kinematics level*, hierachical controllers based on either successive or augmented nullspace projections have been proposed in order to ensure a strict task hierarchy [18, 2]. For the handling of task singularities, a singularity robust inverse kinematics has been proposed [4]. However, this singularity robust inverse destroys the strict task hierarchy and effectively generates a weighting among different tasks.

Other frameworks handle multiple tasks at the *dynamics level*. The operational space approach has been extended in this direction with applications in humanoid robotics [22, 23]. Other Inverse Dynamics (ID) based controllers use hierarchic quadratic programs (QP) [20, 11, 3]. Most of these works aim at a strict task decoupling.

The presented work is inspired by the family of Inverse Dynamics based tracking controllers that softly trade off a set of tasks (collected in a stack of tasks (SoT)) via a single weighted QP [13, 15, 10]. Such controllers are straightforward to write and stand out due to their high flexibility. Yet, compared to passivity-based approaches such as [19, 5, 12, 16, 6], they are less robust w.r.t. modeling errors and contact uncertainties [10, 6]. This causes real-world issues such as vibrations, which are often addressed using heuristic approaches [13, 10].

Furthermore, the weighting based multi-objective controller in [3] and the strictly hierarchical passivity-based controller from [6] served as inspiration for this work. Similar to [3] we use a QP to combine individual control actions from separate task space controllers. However, in [3] each separate control action is computed based on ID with the unit matrix as the desired inertia (feedback linearization). In contrast, the individual task controllers presented here use the concept of passivity and avoid inertia shaping, i.e. we aim at a PD+ like closed-loop for each task [19]. Compared to [6], which also preserves the natural inertia, we use a weighted QP formulation (soft prioritization), which allows us to blend an arbitrary number of different tasks and in certain situations (e.g. when a single task becomes singular) behaves less aggressively.

In this work, we derive a control architecture that is based on nominally passive subtask controllers, the so-called Modular Passive Tracking Controllers (MPTC). These are combined and traded off via a stack of tasks, which is solved via a single weighted pseudo-inverse or QP, respectively. The control framework combines the advantages of both Inverse Dynamics controllers and passivity-based controllers, namely: ease of implementation and use, task space tracking capabilities, passivity and contact robustness, and natural redundancy handling. The corresponding stability analysis is based on Lyapunov theory. For the non-conflicting case, the overall controller is found to be asymptotically stable and passive. An important contribution of the presented work is the derivation of a specific optimization weight that additionally *preserves passivity even in the over-determined* (i.e. conflicting) *case*. For competing tasks and corresponding inconsistent task references, multiple simulations and experiments show evidence of MPTC's stability and robustness even in the tracking case, while a formal stability proof is missing so far.

The paper is organized as follows: Section II derives the Modular Passive Tracking Controller (MPTC) at task level, while section III provides the overall closed-loop analysis and controller derivation. Section IV compares MPTC to Inverse Dynamics (ID) and PD+ based controllers, and presents the wide range of possible decoupling levels. Section V provides simulation and experimental results for both fixed-base and free-floating robots, while section VI concludes the paper.

II. Derivation of Modular Passive Tracking Control (MPTC)

This work considers n_T tasks, each having its own individual objective. To satisfy the single-task objectives, this section derives Modular Passive Tracking Controllers (MPTC), which are combined into different overall controllers in Sec. III.

A. General robot model

The general robot equation of motion can be written as

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{\tau}_g(\boldsymbol{q}) = \boldsymbol{\tau} \quad , \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ denotes the generalized coordinates[1], $\boldsymbol{M}(\boldsymbol{q})$, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ and $\boldsymbol{\tau}_g(\boldsymbol{q})$ are the inertia matrix, Coriolis and centrifugal matrix, and gravitational torques,[2] respectively, and

$$\boldsymbol{\tau} = \boldsymbol{S}^T\,(\boldsymbol{\tau}_j + \boldsymbol{\tau}_{int}) + \underbrace{\boldsymbol{L}_{all}^T\,\boldsymbol{w}_{all}}_{\boldsymbol{\tau}_{ext}} \tag{2}$$

denotes the *generalized forces*. These are composed of joint motor torques $\boldsymbol{\tau}_j$ and internal perturbation torques $\boldsymbol{\tau}_{int}$ acting in the robot joints (e.g. joint friction), which are both mapped to $\boldsymbol{\tau}$ via the joint selection matrix $\boldsymbol{S}$,[3] and of external torques $\boldsymbol{\tau}_{ext}$. The latter are composed of all wrenches $\boldsymbol{w}_{all} = [\boldsymbol{w}_1^T, ..., \boldsymbol{w}_{n_L}^T]^T$ acting on the n_L robot links, which are mapped to $\boldsymbol{\tau}$ via the stack of link Jacobians $\boldsymbol{L}_{all} = [\boldsymbol{L}_1^T, ..., \boldsymbol{L}_{n_L}^T]^T$. While the single elements of (2) will be used in section III-E, in the following we will simply use $\boldsymbol{\tau}$ to represent arbitrary generalized forces.

Solving (1) for the generalized accelerations $\ddot{\boldsymbol{q}}$ yields

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{C}\,\dot{\boldsymbol{q}} - \boldsymbol{\tau}_g\right) \quad . \tag{3}$$

B. Task space quantities

1) Task space velocities and accelerations: In robotic control, for typical task spaces[4] a task velocity vector

$$\dot{\boldsymbol{x}}_k = \boldsymbol{J}_k\,\dot{\boldsymbol{q}} \tag{4}$$

can be formulated. The index k indicates that such a mapping exists for all n_T tasks, i.e. $k \in \{1, .., n_T\}$. Here, $\boldsymbol{J}_k \in \mathbb{R}^{n_{Tk} \times n}$ denotes the corresponding task Jacobian, with the dimensionality $n_{T,k}$ of the k-th task. Differentiating (4) with respect to time and inserting (3), we find the task space acceleration

$$\ddot{\boldsymbol{x}}_k = \dot{\boldsymbol{J}}_k\,\dot{\boldsymbol{q}} + \boldsymbol{J}_k\,\ddot{\boldsymbol{q}} = \boldsymbol{J}_k\,\boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_g\right) - \boldsymbol{Q}_k\,\dot{\boldsymbol{q}} \quad , \tag{5}$$

where

$$\boldsymbol{Q}_k = \boldsymbol{J}_k\,\boldsymbol{M}^{-1}\,\boldsymbol{C} - \dot{\boldsymbol{J}}_k \quad . \tag{6}$$

For the design of a tracking control law (see Sec. II-C), the corresponding task velocity error

$$\dot{\tilde{\boldsymbol{x}}}_k = \dot{\boldsymbol{x}}_{k,ref} - \underbrace{\boldsymbol{J}_k\,\dot{\boldsymbol{q}}}_{\dot{\boldsymbol{x}}_k} \tag{7}$$

and task acceleration error

$$\ddot{\tilde{\boldsymbol{x}}}_k = \ddot{\boldsymbol{x}}_{k,ref} - \underbrace{\left(\boldsymbol{J}_k\,\boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\tau}_g\right) - \boldsymbol{Q}_k\,\dot{\boldsymbol{q}}\right)}_{\ddot{\boldsymbol{x}}_k} \tag{8}$$

are of particular interest. Here, $\dot{\boldsymbol{x}}_{k,ref}$ and $\ddot{\boldsymbol{x}}_{k,ref}$ denote the task reference velocity and acceleration, respectively.

2) Task space inertia and its derivative: Using $\boldsymbol{J}_k$, the inertia matrix $\boldsymbol{M}$ can be projected into the task space [17]:

$$\boldsymbol{M}_k = (\boldsymbol{J}_k\;\boldsymbol{M}^{-1}\;\boldsymbol{J}_k^T)^{-1} \quad . \tag{9}$$

Differentiating (9) yields

$$\begin{aligned}\dot{\boldsymbol{M}}_k &= -\boldsymbol{M}_k\Big(\dot{\boldsymbol{J}}_k\boldsymbol{M}^{-1}\boldsymbol{J}_k^T - \boldsymbol{J}_k\boldsymbol{M}^{-1}\overbrace{\dot{\boldsymbol{M}}}^{\boldsymbol{C}+\boldsymbol{C}^T}\boldsymbol{M}^{-1}\boldsymbol{J}_k^T \\ &\qquad + \boldsymbol{J}_k\boldsymbol{M}^{-1}\dot{\boldsymbol{J}}_k^T\Big)\,\boldsymbol{M}_k \\ &= \boldsymbol{C}_k + \boldsymbol{C}_k^T \quad ,\end{aligned} \tag{10}$$

with the task space Coriolis and centrifugal matrix

$$\boldsymbol{C}_k = \boldsymbol{M}_k\,\boldsymbol{Q}_k\,\boldsymbol{T}_k^T \quad . \tag{11}$$

The matrix

$$\boldsymbol{T}_k = \boldsymbol{M}_k\,\boldsymbol{J}_k\,\boldsymbol{M}^{-1} \tag{12}$$

is the dynamically consistent pseudo-inverse of $\boldsymbol{J}_k^T$.

C. Modular Passive Tracking Controller (MPTC)

This section derives the proposed Modular Passive Tracking Controller (MPTC). It is written in generic form, serving as template for arbitrary specific controllers (e.g. Cartesian or joint controllers). For each one of the n_T tasks, we use a *separate Lyapunov function* based on the task-related relative kinetic energy $E_{kin,k}$ and relative potential energy $E_{pot,k}$:

$$V_k = \underbrace{\frac{1}{2}\,\dot{\tilde{\boldsymbol{x}}}_k^T\,\boldsymbol{M}_k\,\dot{\tilde{\boldsymbol{x}}}_k}_{E_{kin,k}} + \underbrace{\frac{1}{2}\,\tilde{\boldsymbol{x}}_k^T\,\boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k}_{E_{pot,k}} \quad , \tag{13}$$

where the positive definite, symmetric matrix $\boldsymbol{K}_k$ denotes the task stiffness. This Lyapunov function is positive definite in the task position error $\tilde{\boldsymbol{x}}_k$ and the task velocity error $\dot{\tilde{\boldsymbol{x}}}_k$.

Now we differentiate (13) and insert (8), which yields:

$$\begin{aligned}\dot{V}_k &= \dot{\tilde{\boldsymbol{x}}}_k^T\Big(\boldsymbol{M}_k\,\ddot{\tilde{\boldsymbol{x}}}_k + \frac{\dot{\boldsymbol{M}}_k}{2}\,\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k\Big) \\ &= \dot{\tilde{\boldsymbol{x}}}_k^T\Big(\underbrace{\boldsymbol{M}_k\,\boldsymbol{J}_k\,\boldsymbol{M}^{-1}}_{\boldsymbol{T}_k}(\boldsymbol{\tau}_g - \boldsymbol{\tau}) + \boldsymbol{M}_k\,\boldsymbol{Q}_k\,\dot{\boldsymbol{q}} \\ &\qquad + \boldsymbol{M}_k\,\ddot{\boldsymbol{x}}_{k,ref} + \boldsymbol{C}_k\,\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k\Big) \quad .\end{aligned} \tag{14}$$

Here, we made use of the equality

$$\frac{\dot{\boldsymbol{M}}_k}{2} = \frac{\boldsymbol{C}_k^T + \boldsymbol{C}_k}{2} = \underbrace{\frac{\boldsymbol{C}_k^T - \boldsymbol{C}_k}{2}}_{\text{skew-symmetric}} + \boldsymbol{C}_k \quad , \tag{15}$$

which allows us to rewrite $\dot{\tilde{\boldsymbol{x}}}_k^T\,\frac{\dot{\boldsymbol{M}}_k}{2}\,\dot{\tilde{\boldsymbol{x}}}_k$ as $\dot{\tilde{\boldsymbol{x}}}_k^T\,\boldsymbol{C}_k\,\dot{\tilde{\boldsymbol{x}}}_k$, since the skew-symmetric term cancels.

[1] These are simply the joint coordinates $\boldsymbol{q}_j$ in case of fixed-base robots (i.e. $\boldsymbol{q} = \boldsymbol{q}_j$), while additionally containing the robot base coordinates $\boldsymbol{x}_b$ (i.e. $\boldsymbol{q} = [\boldsymbol{x}_b^T, \boldsymbol{q}_j^T]^T$) in case of free-floating robots (e.g. humanoids).

[2] Note: dependencies on $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ will be omitted below.

[3] Note: $\boldsymbol{S}$ is a unit matrix for fixed base robots, while for free-floating robots $\boldsymbol{S} = [\boldsymbol{0}_{n_{act}\times 6}, \boldsymbol{I}_{n_{act}\times n_{act}}]$, where n_{act} denotes the number of actuated robot joints.

[4] Typical tasks are joint level control, Cartesian end effector control, etc.

We now define the (actual) *task force*[5] $\boldsymbol{f}_k$ as:

$$\boldsymbol{f}_k \;=\; \boldsymbol{T}_k\,\boldsymbol{\tau} \quad . \tag{16}$$

By choosing the *desired task force*[6] $\boldsymbol{f}_{k,des}$ as

$$\boldsymbol{f}_{k,des} \;=\; \boldsymbol{T}_k\boldsymbol{\tau}_g + \boldsymbol{M}_k\,\boldsymbol{Q}_k\,\dot{\boldsymbol{q}} + \boldsymbol{M}_k\,\ddot{\boldsymbol{x}}_{k,ref} + (\boldsymbol{C}_k+\boldsymbol{D}_k)\,\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k \tag{17}$$

and rewriting $\boldsymbol{T}_k\,\boldsymbol{\tau}$ as

$$\boldsymbol{T}_k\,\boldsymbol{\tau} = \boldsymbol{f}_k = \boldsymbol{f}_{k,des} - \underbrace{(\boldsymbol{f}_{k,des} - \boldsymbol{f}_k)}_{\tilde{\boldsymbol{f}}_k} \quad , \tag{18}$$

the single task Lyapunov rate from (14) becomes

$$\dot{V}_k \;=\; \underbrace{-\dot{\tilde{\boldsymbol{x}}}_k^T\,\boldsymbol{D}_k\,\dot{\tilde{\boldsymbol{x}}}_k}_{\dot{V}_{k,des},\ \text{purely dissipative}} \;+\; \underbrace{\dot{\tilde{\boldsymbol{x}}}_k^T\,\tilde{\boldsymbol{f}}_k}_{\dot{\tilde{V}}_k} \quad . \tag{19}$$

Note that the controlled system (at task level) is passive with respect to input $\tilde{\boldsymbol{f}}_k$, output $\dot{\tilde{\boldsymbol{x}}}_k$ and the storage function V_k from (13). While the desired Lyapunov rate $\dot{V}_{k,des}$ is purely dissipative for a positive definite damping matrix $\boldsymbol{D}_k$, the term $\dot{\tilde{V}}_k$ may be non-zero, depending on factors including unknown perturbations, under-actuation and other actuation limits, task inconsistencies and prioritization. Finally, we premultiply (8) by $\boldsymbol{M}_k$, insert (18) and (17), simplify and reorder to obtain a task dynamics of the form

$$\boldsymbol{M}_k\,\ddot{\tilde{\boldsymbol{x}}}_k \;+\; \Big(\boldsymbol{C}_k+\boldsymbol{D}_k\Big)\,\dot{\tilde{\boldsymbol{x}}}_k \;+\; \boldsymbol{K}_k\,\tilde{\boldsymbol{x}}_k \;=\; \tilde{\boldsymbol{f}}_k \;. \tag{20}$$

Note that the task related Coriolis term (i.e. $\boldsymbol{C}_k\dot{\tilde{\boldsymbol{x}}}_k$) has not been cancelled, which is a prerequisite for passivity. If the desired task force is achieved (i.e. $\tilde{\boldsymbol{f}}_k = \boldsymbol{0}$), equation (20) corresponds to a spring-mass-damper dynamics for task k. For non-conflicting tasks, one can show asymptotic stability of all trajectories. That can be achieved, for example, by invoking the ε-method [17], to obtain a strong Lyapunov function with negative definite time derivative, similar to [6].

Otherwise, e.g. in case of under-actuation or other actuation limits, unexpected external perturbations or task inconsistencies, (20) corresponds to a compliance[7] behavior [7]. For such cases, further analysis may become necessary. In this work, we focus on the problem of task inconsistencies, which will be addressed in the subsequent section.

III. Overall closed-loop analysis and control

In this section, we derive controllers for the overall system of subtask controllers, i.e. for the complete set of desired task forces $\boldsymbol{f}_{k,des}$ from (17) for $k \in \{1,..,n_T\}$, and analyze their closed-loop behaviors by applying Lyapunov theory.

[5]Note: depending on the task, the task force may contain linear forces, torques, wrenches, etc.

[6]Note: this desired task force $\boldsymbol{f}_{k,des}$ will be used as task-specific controller objective in section III. Also note: in Sec. IV-B, we provide an alternative (yet equivalent) controller formulation (55), which better unveils the controller's similarity to a PD+ formulation.

[7]Remember: compliance means an impedance behavior with natural inertia (no inertia shaping).

A. Definition of different task force errors

Stacking all desired task forces $\boldsymbol{f}_{k,des}$ from (17) for $k \in \{1,..,n_T\}$ yields

$$\boldsymbol{f}_{des} \;=\; \begin{bmatrix} \boldsymbol{f}_{1,des} \\ \vdots \\ \boldsymbol{f}_{n_T,des} \end{bmatrix} \quad . \tag{21}$$

Similarly, we stack (16) for $k \in \{1,..,n_T\}$ to obtain

$$\underbrace{\begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_{n_T} \end{bmatrix}}_{\boldsymbol{f}} \;=\; \underbrace{\begin{bmatrix} \boldsymbol{T}_1 \\ \vdots \\ \boldsymbol{T}_{n_T} \end{bmatrix}}_{\boldsymbol{T}}\,\boldsymbol{\tau} \quad . \tag{22}$$

This is the mapping from the actual generalized forces $\boldsymbol{\tau}$ to the *stack of actual task forces* $\boldsymbol{f}$ via the corresponding collected task mapping matrix $\boldsymbol{T} \in \mathbb{R}^{n_{T,all}\times n}$, where the sum over all subtask dimensionalities is denoted by $n_{T,all} = \sum_{k=1}^{n_T} n_{T,k}$. Now, we define the *stack of actual task force errors*

$$\tilde{\boldsymbol{f}} \;=\; \boldsymbol{f}_{des} - \boldsymbol{f} \;=\; \boldsymbol{f}_{des} - \boldsymbol{T}\boldsymbol{\tau} \quad . \tag{23}$$

Next, to facilitate discussions about *commanded task forces* and corresponding errors, we evaluate (22) for the *commanded* generalized forces $\boldsymbol{\tau}_{cmd}$ from some controller[8], which yields the stack of commanded task forces

$$\boldsymbol{f}_{cmd} \;=\; \begin{bmatrix} \boldsymbol{f}_{1,cmd} \\ \vdots \\ \boldsymbol{f}_{n_T,cmd} \end{bmatrix} \;=\; \boldsymbol{T}\;\boldsymbol{\tau}_{cmd} \;=\; \underbrace{\boldsymbol{T}\,\boldsymbol{U}}_{\boldsymbol{T}_u}\,\boldsymbol{u}_{cmd} \quad , \tag{24}$$

For certain control problems (including under-actuation, see Sec. III-E), $\boldsymbol{\tau}_{cmd}$ may result from the given optimization variables $\boldsymbol{u}_{cmd}$, which are mapped to corresponding generalized forces via the actuation mapping matrix $\boldsymbol{U}$, i.e. $\boldsymbol{\tau}_{cmd} = \boldsymbol{U}\,\boldsymbol{u}_{cmd}$.

Finally, subtracting (24) from (21), we obtain the corresponding stack of task force command errors:

$$\tilde{\boldsymbol{f}}_{cmd} \;=\; \boldsymbol{f}_{des} - \boldsymbol{f}_{cmd} \;=\; \boldsymbol{f}_{des} - \boldsymbol{T}\,\boldsymbol{\tau}_{cmd} \;=\; \boldsymbol{f}_{des} - \boldsymbol{T}_u\,\boldsymbol{u}_{cmd} \quad . \tag{25}$$

B. Definition of overall system Lyapunov function, and its actual and commanded derivatives

In this section, we will analyze the stability and passivity of *complete sets of modular task space controllers* (and thus the stability of the complete robot system dynamics, if all robot DOF are covered by the collection of task coordinates). To this end, we combine all single task Lyapunov functions (13) to construct the following overall Lyapunov function

$$V \;=\; \sum_{k=1}^{n_T} \Big(\psi_k\,V_k\Big) \quad , \tag{26}$$

[8]Possible controllers may, for example, use pseudo-inverse based optimization as in (36) for unconstrained control problems or quadratic programming (QP) based optimization (see Sec. III-E) to handle inequality constraints.

which for positive scalar weights[9] $\psi_k > 0$ is positive definite (just like its input elements V_k). Correspondingly, the overall Lyapunov function derivative $\dot{V}$ is obtained by combining the single task Lyapunov derivatives $\dot{V}_k$ from (19):

$$\dot{V} = \sum_{k=1}^{n_T} \left(\psi_k \dot{V}_k \right) = \underbrace{\sum_{k=1}^{n_T} \left(\psi_k \dot{V}_{k,des} \right)}_{\dot{V}_{des}} + \underbrace{\sum_{k=1}^{n_T} \left(\psi_k \dot{\tilde{V}}_k \right)}_{\dot{\tilde{V}}} \quad . \quad (27)$$

The term $\dot{V}_{des}$ is negative semi-definite if all $\psi_k > 0$, since all desired task Lyapunov rates $\dot{V}_{k,des}$ are negative semi-definite. The *actual* overall Lyapunov rate error $\dot{\tilde{V}}$ can be written as

$$\dot{\tilde{V}} = \dot{\tilde{\boldsymbol{x}}}^T \underbrace{\begin{bmatrix} \psi_1 \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \psi_{n_T} \boldsymbol{I} \end{bmatrix}}_{\boldsymbol{\Psi}} \tilde{\boldsymbol{f}} \quad , \quad (28)$$

where $\dot{\tilde{\boldsymbol{x}}}^T = \left[\dot{\tilde{\boldsymbol{x}}}_1^T, ..., \dot{\tilde{\boldsymbol{x}}}_{n_T}^T \right]$. Note: $\dot{\tilde{V}}$ is a function of $\tilde{\boldsymbol{f}}$, and thus also of the actual generalized forces $\boldsymbol{\tau}$.

We will now additionally examine the effect of the stack of *commanded* task force errors $\tilde{\boldsymbol{f}}_{cmd}$. Adapting (28) correspondingly, we find the *commanded* overall Lyapunov rate error

$$\dot{\tilde{V}}_{cmd} = \dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Psi} \tilde{\boldsymbol{f}}_{cmd} \quad , \quad (29)$$

which corresponds to the commanded overall Lyapunov rate

$$\dot{V}_{cmd} = \dot{V}_{des} + \underbrace{\sum_{k=1}^{n_T} \left(\psi_k \dot{\tilde{V}}_{k,cmd} \right)}_{\dot{\tilde{V}}_{cmd}} \quad , \quad (30)$$

where $\dot{\tilde{V}}_{k,cmd} = \dot{\tilde{\boldsymbol{x}}}_k^T \tilde{\boldsymbol{f}}_{k,cmd}$. Finally, we reformulate (28) as

$$\begin{aligned} \dot{\tilde{V}} &= \dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Psi} \left(\overbrace{\boldsymbol{f}_{des} - \boldsymbol{f}}^{\tilde{\boldsymbol{f}}} \right) \quad (31) \\ &= \underbrace{\dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Psi} \left(\overbrace{\boldsymbol{f}_{des} - \boldsymbol{f}_{cmd}}^{\tilde{\boldsymbol{f}}_{cmd}} \right)}_{\dot{\tilde{V}}_{cmd}} + \underbrace{\dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Psi} \left(\overbrace{\boldsymbol{f}_{cmd} - \boldsymbol{f}}^{\tilde{\boldsymbol{f}}_{real}} \right)}_{\dot{\tilde{V}}_{real}} \quad , \end{aligned}$$

which corresponds to the overall *actual Lyapunov derivative*

$$\dot{V} = \dot{V}_{des} + \dot{\tilde{V}}_{cmd} + \dot{\tilde{V}}_{real} \quad (32)$$

that relates to the *real* system behavior. Here,

$$\dot{\tilde{V}}_{real} = \dot{\tilde{\boldsymbol{x}}}^T \boldsymbol{\Psi} \left(\overbrace{\boldsymbol{f}_{cmd} - \boldsymbol{f}}^{\tilde{\boldsymbol{f}}_{real}} \right) \quad (33)$$

denotes the component of the Lyapunov rate error that corresponds to deviations $\tilde{\boldsymbol{f}}_{real}$ of the commanded task forces $\boldsymbol{f}_{cmd}$ from the actual ones $\boldsymbol{f}$.

[9]Note: These positive scalar weights ψ_k are equivalent to the optimization weights used in (34), (41).

C. Overall cost function

Based on $\tilde{\boldsymbol{f}}_{cmd}$ from (25), we formulate an overall cost function

$$\begin{aligned} G &= \frac{1}{2} \tilde{\boldsymbol{f}}_{cmd}^T \boldsymbol{W} \tilde{\boldsymbol{f}}_{cmd} \quad (34) \\ &= \frac{1}{2} \boldsymbol{\tau}_{cmd}^T \boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \boldsymbol{\tau}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T} \boldsymbol{\tau}_{cmd} + \frac{1}{2} \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{f}_{des} \\ &= \frac{1}{2} \boldsymbol{u}_{cmd}^T \boldsymbol{T}_u^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} + \frac{1}{2} \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{f}_{des} \quad . \end{aligned}$$

Here, $\boldsymbol{W}$ denotes an arbitrary symmetric and positive-definite weighting matrix. This cost function G will be minimized by the different controllers presented in sections III-D and III-E.

D. Unconstrained and fully actuated case

1) Pseudo-inverse based general analytical optimization: This section considers the fully actuated, unconstrained and potentially conflicting case. The lack of inequality constraints facilitates an analytical solution via weighted pseudo-inverse, while the full actuation guarantees controllability and allows to directly use the commanded generalized forces $\boldsymbol{\tau}_{cmd}$ as optimization variables. In this case, to optimize the cost function (34, second line), we differentiate G w.r.t. $\boldsymbol{\tau}_{cmd}$:

$$\frac{dG}{d\boldsymbol{\tau}_{cmd}} = \boldsymbol{\tau}_{cmd}^T \boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T} \quad . \quad (35)$$

The cost function (34) is minimized by setting (35) to zero and solving for the controller torques $\boldsymbol{\tau}_{cmd}$

$$\boldsymbol{\tau}_{cmd} = \left(\boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \right)^{-1} \boldsymbol{T}^T \boldsymbol{W} \boldsymbol{f}_{des} \quad , \quad (36)$$

which are the optimal torque commands for the given problem statement. Inserting (36) in (25)

$$\tilde{\boldsymbol{f}}_{cmd} = \underbrace{\left(\boldsymbol{I} - \boldsymbol{T} \left(\boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \right)^{-1} \boldsymbol{T}^T \boldsymbol{W} \right)}_{\boldsymbol{E}_T} \boldsymbol{f}_{des} \quad , \quad (37)$$

where $\boldsymbol{E}_T$ denotes the *task force trade-off matrix*.

2) Non-conflicting case: If $\boldsymbol{T}$ is square and invertible (i.e. all subtasks are independent from each other and thus non-conflicting), the trade-off matrix from (37) becomes $\boldsymbol{E}_T = \boldsymbol{0}$, and the task force command errors become $\tilde{\boldsymbol{f}}_{cmd} = \boldsymbol{0}$. In this case, $\dot{\tilde{V}}_{cmd}$ from (29) is also zero, and thus $\dot{V}$ from (32) becomes

$$\dot{V} = \dot{V}_{des} + \dot{\tilde{V}}_{real} \quad . \quad (38)$$

3) Conflicting case: If, in contrast, $\boldsymbol{T}$ is non-invertible (e.g. in over-determined controller setups), the trade-off matrix $\boldsymbol{E}_T$ is non-zero. Inserting (37) into (29), we find the corresponding commanded overall Lyapunov rate error

$$\dot{\tilde{V}}_{cmd} = (\dot{\boldsymbol{x}}_{ref}^T - \dot{\boldsymbol{q}}^T \boldsymbol{J}^T) \boldsymbol{\Psi} \boldsymbol{E}_T \boldsymbol{f}_{des} \quad . \quad (39)$$

Here, $\boldsymbol{J} = [\boldsymbol{J}_1^T, ..., \boldsymbol{J}_{n_T}^T]^T$ is a stack of single task Jacobians.

Considering the *conflicting regulation case* (i.e. $\dot{\boldsymbol{x}}_{ref} = \boldsymbol{0}$) first, we examine the matrix $\boldsymbol{J}^T \boldsymbol{\Psi} \boldsymbol{E}_T$, which appears in (39):

$$\begin{aligned} \boldsymbol{J}^T \boldsymbol{\Psi} \boldsymbol{E}_T &= \boldsymbol{J}^T \boldsymbol{\Psi} \left(\boldsymbol{I} - \boldsymbol{T} \left(\boldsymbol{T}^T \boldsymbol{W} \boldsymbol{T} \right)^{-1} \boldsymbol{T}^T \boldsymbol{W} \right) \quad (40) \\ &= \boldsymbol{J}^T \boldsymbol{\Psi} \left(\boldsymbol{I} - \boldsymbol{\Lambda} \boldsymbol{J} \left(\boldsymbol{J}^T \boldsymbol{\Lambda} \boldsymbol{W} \boldsymbol{\Lambda} \boldsymbol{J} \right)^{-1} \boldsymbol{J}^T \boldsymbol{\Lambda} \boldsymbol{W} \right). \end{aligned}$$

Here, we inserted $\boldsymbol{T} = \boldsymbol{\Lambda}\, \boldsymbol{J}\, \boldsymbol{M}^{-1}$, where $\boldsymbol{\Lambda}$ is a block-diagonal matrix with the task space inertia matrices $\{\boldsymbol{M}_1, \ldots, \boldsymbol{M}_{n_r}\}$ as its diagonal submatrices. For an arbitrary choice of $\boldsymbol{W}$,[10] (40) is a non-zero matrix, such that $\dot{\tilde{V}}_{cmd}$ from (39) is non-zero, even in the regulation case. However, for the choice

$$\boldsymbol{W} \;=\; \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi} \tag{41}$$

equation (40) becomes

$$\boldsymbol{J}^T \boldsymbol{\Psi}\, \boldsymbol{E}_T = \boldsymbol{J}^T \boldsymbol{\Psi} - \underbrace{\boldsymbol{J}^T \boldsymbol{\Psi}\, \boldsymbol{\Lambda}\, \boldsymbol{J} \Big(\boldsymbol{J}^T \boldsymbol{\Psi}\, \boldsymbol{\Lambda}\, \boldsymbol{J} \Big)^{-1}}_{\boldsymbol{I}} \boldsymbol{J}^T \boldsymbol{\Psi} = \boldsymbol{0}. \tag{42}$$

This means that (independently from the current generalized velocities $\dot{\boldsymbol{q}}$ and desired task forces $\boldsymbol{f}_{des}$) for the choice (41) the commanded Lyapunov rate $\dot{\tilde{V}}_{cmd}$ from (39) becomes

$$\dot{\tilde{V}}_{cmd} = 0 \tag{43}$$

in the regulation case ($\dot{\boldsymbol{x}}_{ref} = \boldsymbol{0}$), and $\dot{V}$ from (32) becomes

$$\dot{V} \;=\; \dot{V}_{des} + \dot{\tilde{V}}_{real} \;. \tag{44}$$

Looking at the elements of (44), we find that the overall controlled system is passive with respect to the input $\boldsymbol{\Psi}\, \tilde{\boldsymbol{f}}_{real}$, the output $\dot{\tilde{\boldsymbol{x}}}$ (these two being the elements of $\dot{\tilde{V}}_{real}$) and the positive definite storage function V from (26). Thus, we conclude passivity of the overall system in the conflicting regulation case examined here. For the overall system closed-loop dynamics of *over-determined / conflicting task setups* (regulation case), (41) acts as *Passivity Warranting Optimization Weight* (PWOW). Note: (43) does not mean that the single task Lyapunov rate errors $\dot{\tilde{V}}_{k,cmd}$ are zero (only their sum).

The overall weighting matrix $\boldsymbol{W} = \boldsymbol{\Lambda}^{-1} \boldsymbol{\Psi}$ from (41) is symmetric and block-diagonal. Its symmetry property results from the symmetry of its diagonal sub-matrices

$$\boldsymbol{W}_k \;=\; \psi_k\, \boldsymbol{M}_k^{-1} \quad , \tag{45}$$

which are a function of ψ_k and the inverse of the (symmetric) task inertia matrix $\boldsymbol{M}_k$. Looking at (45), it becomes clear that each task can still be weighted independently (w.r.t. other tasks) via its corresponding weighting scalar ψ_k.

Now, we consider the *conflicting tracking case*. For $\dot{\boldsymbol{x}}_{ref} \neq \boldsymbol{0}$ and still applying (41), (39) turns into

$$\dot{\tilde{V}}_{cmd} \;=\; \dot{\boldsymbol{x}}_{ref}^T\, \boldsymbol{\Psi}\, \boldsymbol{E}_T \;\; \boldsymbol{f}_{des} \;, \tag{46}$$

for which equation (32) becomes

$$\dot{V} \;=\; \dot{V}_{des} + \underbrace{\dot{\boldsymbol{x}}_{ref}^T\, \boldsymbol{\Psi}\, \boldsymbol{E}_T \;\; \boldsymbol{f}_{des}}_{\dot{\tilde{V}}_{cmd}} + \dot{\tilde{V}}_{real} \quad . \tag{47}$$

For *inconsistent* task reference velocities, $\dot{\tilde{V}}_{cmd}$ will typically be non-zero, which renders a formal passivity proof for the tracking case more difficult (out of the scope of this paper).

[10] even if it is chosen to be diagonal, as often found in the Inverse Dynamics (ID) related literature

E. Handling under-actuation and other actuation constraints

The analytical solutions presented in the previous section are dedicated to control problems that assume fully actuated robots, whose actuation limits (or other constraints) are not relevant. This section will treat the cases of under-actuation and actuation constraints, and propose a solution for such control problems in the MPTC context.

1) Under-actuation: In contrast to fixed-base robots, the robot base of free-floating robots (e.g. humanoids) is not actuated. Instead, in order to achieve a certain degree of controllability, a free-floating robot needs to use its end effectors to create contact wrenches that compensate for the lack of a direct base actuation. The corresponding actuation mapping matrix $\boldsymbol{U}$ from Sec. III-A has the following form:

$$\boldsymbol{U} = [\boldsymbol{S}^T, \boldsymbol{L}_{EE}^T] \;. \tag{48}$$

The related actuation DOF / optimization variables are

$$\boldsymbol{u}_{cmd} = \begin{bmatrix} \boldsymbol{\tau}_{j,cmd} \\ \boldsymbol{w}_{EE,cmd} \end{bmatrix} \;, \tag{49}$$

where $\boldsymbol{\tau}_{j,cmd}$ denotes the commanded joint torques and $\boldsymbol{w}_{EE,cmd}$ are the commanded end effector wrenches. Equations (48) and (49) are based on elements from equation (2). Note however that here we replaced the collection of all link Jacobians $\boldsymbol{L}_{all}$ and link wrenches $\boldsymbol{w}_{all}$ by a selection that corresponds to the contacting end effectors ("EE"), i.e. by $\boldsymbol{L}_{EE}$ and $\boldsymbol{w}_{EE,cmd}$.

2) contact and actuation constraints: The commanded end effector wrenches $\boldsymbol{w}_{EE,cmd}$ just introduced are often subject to inequality constraints (the so-called "contact constraints"). As an example, in walking related applications (see Sec. V-B) these contact constraints are typically expressed in the form of unilaterality and friction cone constraints. Omitting such contact constraints may lead to a failure of the robot. Additionally, due to the physical limitations of the robot, it often makes sense to also constrain the commanded joint torques $\boldsymbol{\tau}_{j,cmd}$. This way, actuator saturation may be avoided.

3) solution via quadratic programming (QP): A popular method that allows us to handle the mentioned problems of under-actuation, actuation limits and contact constraints is to set up a quadratic program (QP) of the form[11]

$$\begin{aligned} \underset{\boldsymbol{u}_{cmd}}{\text{minimize}} \quad & G_{QP} = \frac{1}{2} \boldsymbol{u}_{cmd}^T \boldsymbol{T}_u^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} - \boldsymbol{f}_{des}^T \boldsymbol{W} \boldsymbol{T}_u \boldsymbol{u}_{cmd} \quad , \\ \text{subject to} \quad & \text{contact and joint torque constraints} \quad . \end{aligned} \tag{50}$$

Here, the third line of (34) was adapted for the formulation of the QP cost function G_{QP}. It is well-known, that a QP provides the same solution as a weighted pseudo-inverse based optimization, as long the constraints are not active. We propose to use the exact same Passivity Warranting Optimization Weight (PWOW) that was applied in Sec. III-D for the analytical optimization, i.e. $\boldsymbol{W} = \boldsymbol{\Lambda}^{-1}\, \boldsymbol{\Psi}$. Note however that, as soon as the constraints become active, passivity can no longer be guaranteed; even in the regulation case.

[11] More explicit QP constraint formulations can be found e.g. in [21, 15, 10].

IV. Placement among related controller types and granularity of targeted decoupling

A. Comparison of Inverse Dynamics (ID) and MPTC

In this section, we compare the objectives of Inverse Dynamics (ID) based controllers and MPTC-based controllers, and show major similarities.

Inverse Dynamics (ID):
Inverse Dynamics (ID) based controllers typically implement a stable second order dynamics of the form

$$\ddot{\boldsymbol{x}}_k = \ddot{\boldsymbol{x}}_{k,ref} + \hat{\boldsymbol{D}}_{ID,k}\,(\underbrace{\dot{\boldsymbol{x}}_{k,ref} - \dot{\boldsymbol{x}}_k}_{\dot{\tilde{\boldsymbol{x}}}_k}) + \hat{\boldsymbol{P}}_{ID,k}\,(\underbrace{\boldsymbol{x}_{k,ref} - \boldsymbol{x}_k}_{\tilde{\boldsymbol{x}}_k}) \;, \quad (51)$$

which enables the tracking of the reference motion $\boldsymbol{x}_{k,ref}$, $\dot{\boldsymbol{x}}_{k,ref}$, $\ddot{\boldsymbol{x}}_{k,ref}$ for positive definite proportional and derivative gain matrices $\hat{\boldsymbol{P}}_{ID,k}$ and $\hat{\boldsymbol{D}}_{ID,k}$. These two gain matrices are typically designed as diagonal matrices in order to obtain a fully decoupled, linear dynamics. Oftentimes, pole placement [1] is used to achieve a critically damped transient response. Inserting the task space acceleration $\ddot{\boldsymbol{x}}_k$ from (5) into (51), and solving for the term related to the generalized forces $\boldsymbol{\tau}$ yields:

$$\begin{aligned}\boldsymbol{J}_k\,\boldsymbol{M}^{-1}\,\boldsymbol{\tau} \;=\; & \boldsymbol{J}_k\,\boldsymbol{M}^{-1}\,\boldsymbol{\tau}_g + \boldsymbol{Q}_k\,\dot{\boldsymbol{q}} \\ & + \ddot{\boldsymbol{x}}_{k,ref} + \hat{\boldsymbol{D}}_{ID,k}\,\dot{\tilde{\boldsymbol{x}}}_k + \hat{\boldsymbol{P}}_{ID,k}\,\tilde{\boldsymbol{x}}_k \quad .\end{aligned} \quad (52)$$

Note that we do not solve for $\boldsymbol{\tau}$ directly here, since, depending on the chosen controller setup, $\boldsymbol{J}_k$ may not be invertible.

Modular Passive Tracking Control (MPTC):
By replacing $\boldsymbol{f}_{k,des}$ in (17) by $\boldsymbol{f}_k = \boldsymbol{T}_k\,\boldsymbol{\tau}$ (from (16)), premultiplying the result with $\boldsymbol{M}_k^{-1}$ and reordering, we find the following desired task control action:

$$\begin{aligned}\boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{\tau} \;=\; & \boldsymbol{J}_k\boldsymbol{M}^{-1}\boldsymbol{\tau}_g + \boldsymbol{Q}_k\dot{\boldsymbol{q}} \\ & + \ddot{\boldsymbol{x}}_{k,ref} + \underbrace{\boldsymbol{M}_k^{-1}(\boldsymbol{C}_k+\boldsymbol{D}_k)}_{\hat{\boldsymbol{D}}_{MPTC,k}}\dot{\tilde{\boldsymbol{x}}}_k + \underbrace{\boldsymbol{M}_k^{-1}\boldsymbol{K}_k}_{\hat{\boldsymbol{P}}_{MPTC,k}}\tilde{\boldsymbol{x}}_k.\end{aligned} \quad (53)$$

When comparing (53) to (52), we find that the basic structure of the presented Modular Passive Tracking Controller (MPTC) and Inverse Dynamics (ID) based controllers is the same. They only differ by the chosen proportional and derivative gain matrices. For ID based controllers, $\hat{\boldsymbol{P}}_{ID,k}$ and $\hat{\boldsymbol{D}}_{ID,k}$ are designed to be constant, whereas, in case of MPTC, $\hat{\boldsymbol{P}}_{MPTC,k}$ and $\hat{\boldsymbol{D}}_{MPTC,k}$ are a function of the current task inertia $\boldsymbol{M}_k$. Additionally, $\hat{\boldsymbol{D}}_{MPTC,k}$ contains the task Coriolis matrix $\boldsymbol{C}_k$, such that the task related Coriolis terms are not canceled, which is a prerequisite for passivity and increases robustness[12].

It is important to note, that (52) and (53) are denoted in task *acceleration* space, where for ID we find constant gain matrices, which correspond to constant system/task eigenvalues, while the local eigenvalues of MPTC are configuration-dependent. If, in contrast, (52) and (53) are transferred into task force space by premultiplying them with $\boldsymbol{M}_k$, we find that the perceived damping and stiffness are constant in case of MPTC, while being configuration dependent for ID.

[12]This is due to the reduced number of feedback channels that may cause problems related to sensor noise and modeling errors.

Remark: The strong similarity between (52) and (53) may be advantageous with regard to a potential porting of existing ID-based controller frameworks to the MPTC methodology.

B. Comparison to PD+ control / passivity-based WBC

Reordering those terms in the subtask controller objective (17) that are related to Coriolis and centrifugal (from here on abbreviated as "CC") effects, we find the following equality:

$$\boldsymbol{C}_k\,\dot{\tilde{\boldsymbol{x}}}_k \;+\; \boldsymbol{M}_k\,\boldsymbol{Q}_k\,\dot{\boldsymbol{q}} \;=\; \boldsymbol{C}_k\,\dot{\boldsymbol{x}}_{k,ref} + \underbrace{\boldsymbol{M}_k\,\boldsymbol{Q}_k\,\Big(\underbrace{\boldsymbol{I} - \boldsymbol{T}_k^T\,\boldsymbol{J}_k}_{\boldsymbol{N}_k}\Big)}_{\boldsymbol{B}_k}\,\dot{\boldsymbol{q}} \quad . \quad (54)$$

Here, $\boldsymbol{N}_k$ denotes a nullspace projector that cancels out all components of $\dot{\boldsymbol{q}}$ that would have an effect on the task-space velocity $\dot{\boldsymbol{x}}_k$, and the matrix $\boldsymbol{B}_k$ collects the corresponding CC related feedback terms. Inserting (54) in (17), we find an alternative (yet equivalent) subtask controller formulation:

$$\boldsymbol{f}_{k,des} = \boldsymbol{T}_k\boldsymbol{\tau}_g + \boldsymbol{B}_k\dot{\boldsymbol{q}} + \boldsymbol{M}_k\ddot{\boldsymbol{x}}_{k,ref} + \boldsymbol{C}_k\dot{\boldsymbol{x}}_{k,ref} + \boldsymbol{D}_k\dot{\tilde{\boldsymbol{x}}}_k + \boldsymbol{K}_k\tilde{\boldsymbol{x}}_k. \quad (55)$$

As compared to the original subtask controller formulation from (17), this formulation enables a more straightforward insight into the CC term cancellation policy of the MPTC controller: MPTC cancels out all *non-task-related CC feedback terms* (comparable to off-diagonal terms in [7]) and provides a task-related feedforward term $\boldsymbol{C}_k\dot{\boldsymbol{x}}_{k,ref}$, while not canceling/utilizing task-related CC feedback terms. The latter can also be verified by revisiting the nominal subtask closed-loop behavior (20), in which the task-related CC terms remain unaffected. Maintaining the task-related CC terms is a prerequisite for passivity, and leads to a higher robustness against modeling errors as compared to Inverse Dynamics based controllers. The latter completely *cancel all CC effects* (based on the potentially inaccurate robot model), while MPTC-based controllers cancel only the non-task-related CC terms.

An interesting special case is, when only one task is considered (combining all subtasks into an overall task by stacking the corresponding subtask Jacobians) and the task Jacobian $\boldsymbol{J}_k$ is square and invertible. This requires the subtasks to be non-conflicting. In this particular case, $\boldsymbol{B}_k = \boldsymbol{0}$ and thus (55) is equivalent to a classic PD+ controller [19] (see Fig. 1) or (in case of inequality constraints) to more recently introduced passivity-based whole-body controllers (WBC) [12, 16].

C. Granularity of targeted decoupling

MPTC is a generic controller design tool, both w.r.t. the choice of sub-controller types (e.g. Cartesian vs. joint control) and w.r.t. the "granularity" of the targeted decoupling[13]. When speaking about subtasks, we never specified their respective dimensions, i.e. the number of DOF that each task covers. This

[13]Note: we are talking here about "targeted decoupling". In case of consistent tasks, MPTC yields a strictly decoupled error dynamics. However, in case of inconsistencies, a coupling between the respective task error dynamics is inevitable (at least for soft task prioritization).

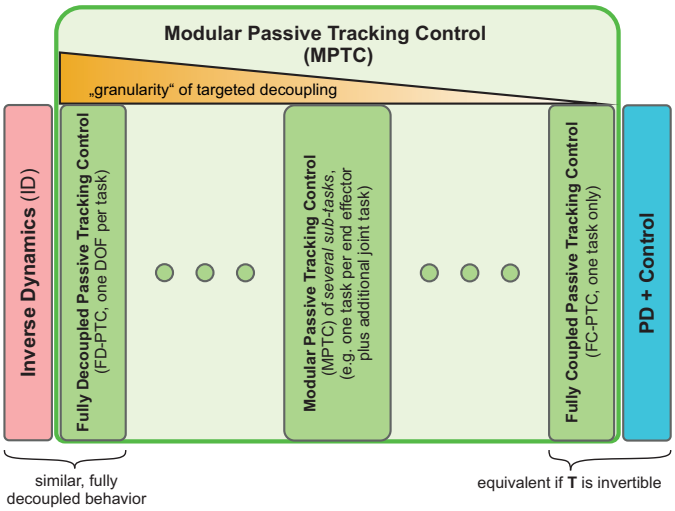

Figure 1: MPTC covers the complete range between Inverse Dynamics (ID) and PD+ Control.

means there are many conceivable controller configurations. Collecting different control objectives (e.g. a Cartesian end effector tracking objective and a joint control one) into a single stack of tasks (SoT), the question remains open, which parts of the SoT should be assigned to what MPTC-related subtasks. As an example: a six-DOF Cartesian task can be defined as two decoupled (linear and angular) controller subtasks, or alternatively as a single six-DOF task. In the first case, the linear and angular error dynamics would be decoupled from each other, while in the second case they would be coupled.

One may also decide to combine all control objectives into a single task (resulting in a *Fully Coupled Passive Tracking Controller* (FC-PTC)), which under certain circumstances is equivalent to a PD+ controller (see Sec. IV-B and Fig. 1). Alternatively, one may define a set of multiple natural tasks (e.g. one task for the left foot, one for the right), which can be treated by the standard MPTC framework. Finally, one may decide to design a maximally granular controller setup, in which each line of the SoT forms a single task. We refer to this particular type of controller setup by Fully Decoupled Passive Tracking Controller (FD-PTC). Out of all possible MPTC setups, FD-PTC is the one that is most similar to an Inverse Dynamics controller; both are based on single-DOF decoupled task dynamics (see Fig. 1). However, FD-PTC can be expected to behave more robustly due to the absence of inertia shaping and reduced CC term cancellation.

Remember that there is only one single weighting scalar ψ_k per task, that allows for (softly) prioritizing the tasks amongst each other. This decreases the required tuning efforts. Note however, that to independently increase the priority of a certain task component (e.g. prefer the z-direction over the x and y components of a Cartesian task), it needs to be assigned to an *own task* and an appropriate weight (e.g. ψ_z).

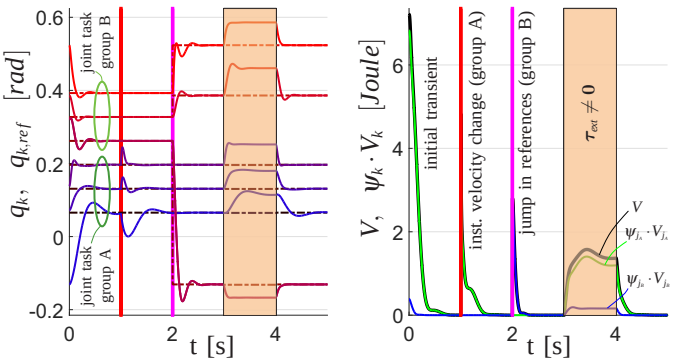

Figure 2: Transient responses for fully determined and non-conflicting task setup. Left: joint angles and references. Right: overall and task-specific Lyapunov function values.

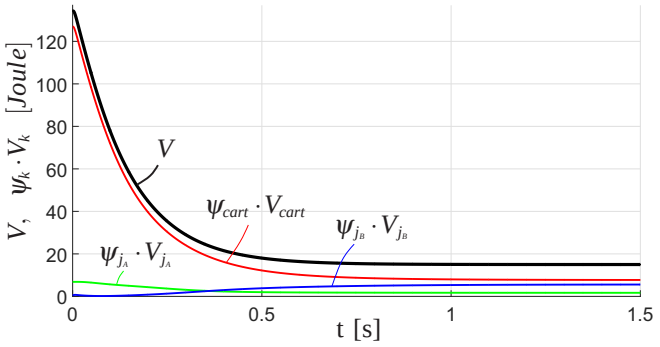

Figure 3: Transient response for over-determined (and thus conflicting) task setup.

V. Simulations and Experiments

In order to validate the performance of our proposed MPTC framework, we performed several simulations and experiments: on the one hand simple simulations based on a forward integration of (3) that used the computed controller torques as input, on the other hand simulations (using OpenHRP [14] as simulation environment) and experiments with the full-sized humanoid robot TORO [8].

A. Fixed-base robot simulations

The first presented simulation is designed to examine the regulation case for a fully actuated, fully determined and thus non-conflicting task setup. To this end, the six joints of a fixed-base robot arm (whose kinematics and inertia properties correspond to one of TORO's legs) were assigned to two different joint tasks A and B (each covering three joints, see Fig. 2). The corresponding joint task stiffness and damping gains[14] were chosen as $\boldsymbol{K}_A = 300\boldsymbol{I}_{3\times3}$, $\boldsymbol{D}_A = diag([40,20,10])$, $\boldsymbol{K}_B = 40\boldsymbol{I}_{3\times3}$ and $\boldsymbol{D}_B = 2\boldsymbol{I}_{3\times3}$, respectively. Note that the scalar task weights ψ_k (here all set to 1) are without effect, since due to the lack of conflicts all tasks were perfectly fulfilled anyway. It has to be noted, that the controller did

[14]Note: for brevity, the units for stiffness (linear: $\frac{N}{m}$, angular: $\frac{Nm}{rad}$) and damping (linear: $\frac{Ns}{m}$, angular: $\frac{Nms}{rad}$) are omitted here.

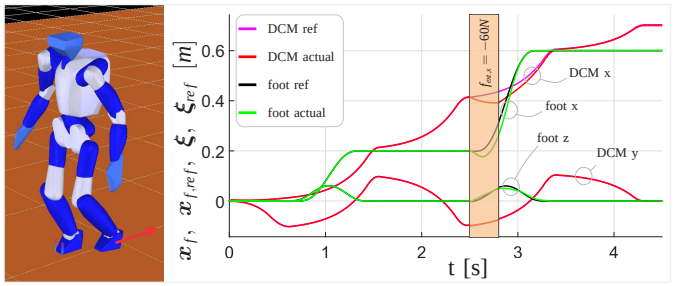

Figure 4: Humanoid robot TORO walking in OpenHRP simulation (single and double support times: $T_{SS} = 0.8s$, $T_{DS} = 0.12s$). After 2.5s, the left foot is perturbed by an external force of $-60N$ in x-direction for 0.3s.

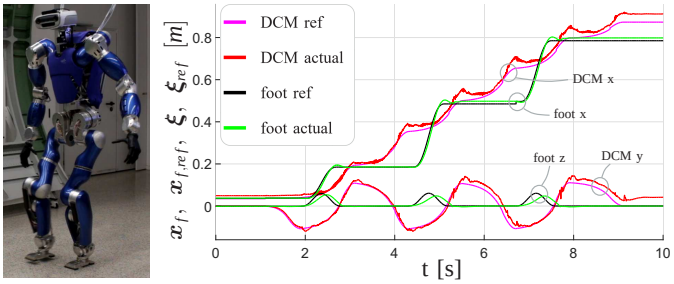

Figure 5: TORO walking in experiment (single and double support times: $T_{SS} = 0.9s$, $T_{DS} = 0.3s$, step length 0.15m).

not require any additional tuning. The presented parameters were chosen to improve the educational value of figure 2. At the beginning of the simulation, all joints rapidly converge to the initial setpoints. After one second, the joints from group A were subject to a velocity change, resulting in a jump in the corresponding task error derivatives. Note that, while group A converges back, group B is completely unaffected due to the task decoupling. After two seconds, the setpoints of group B are changed. Again, the corresponding joint coordinates converge, while now group A is completely unaffected. Note: the perfect task decoupling observed here was only achievable due to the consistency of the two joint tasks. Finally, after three seconds a torque offset of $\tau_{ext} = [20, 20, 20, 5, 5, 5]Nm$ is applied first and then removed at 4 s. Again the observed controller behavior is good-natured and fulfills our expectations. Using this simulation, we can verify that the overall Lyapunov function value V always decreases, except for the case of perturbations and setpoint changes (see Fig. 2 (right)).

The second presented simulation evaluates the controller's performance for a conflicting task setup, again for the regulation case. This time, a six-DOF Cartesian end effector task with stiffness $\boldsymbol{K}_{cart} = diag([2000, 2000, 2000, 100, 100, 100])$ and damping $\boldsymbol{D}_{cart} = diag([500, 500, 500, 20, 20, 20])$ was added to the previously described joint tasks A and B. The scalar task weights ψ_k are set to 1. After starting the controller, the robot converges to an equilibrium position. Note that the overall Lyapunov function value V decreases monotonically, while the weighted subtask Lyapunov function values may also grow (as for example $\psi_{j_8} \cdot V_{j_8}$ in Fig. 3).

B. Humanoid robot simulations and experiments

We performed walking simulations of the humanoid robot TORO (see Fig. 4), which were based on a hybrid WBC setup: Inverse Dynamics based torso orientation and overall posture tasks, a task for Divergent Component of Motion (DCM) [9] control and angular momentum regularization, and Cartesian (6-DOF) MPTC-based controllers for foot tracking were mixed[15]. Precise tracking of the foot reference trajectories is achieved. After 2.5s, the left foot is perturbed by a force of $-60N$ in x-direction for 0.3s, resulting in a maximum foot position error of 51mm and a maximum DCM error of 45mm. After the perturbation is removed, the foot converges fast enough to successfully continue walking.

To evaluate the real-world performance of MPTC, we conducted several experiments using TORO, including push recovery (during stance), human-robot interaction and walking (a video can be found here https://youtu.be/WdF9UQK8aIo). Here, we present a walking experiment during which TORO took six steps forward (see Fig. 5 for details). *Separate* MPTC-based controllers were used for controlling the feet (six-DOF Cartesian tracking), torso (three-DOF rotational tracking), leg joints, waist joint and upper body joints (the latter three serving as overall pose control and regularization tasks). As for the simulations, a DCM-based controller and an angular momentum regularization controller were additionally applied. The walking performance was overall robust. However, tracking errors can be observed, which we believe are caused by torque offsets and joint friction. As compared to ID-based controllers [10], the tuning effort was low, giving evidence to MPTC's robustness in real-world settings.

VI. Conclusion and Future Work

This work introduced the so-called Modular Passive Tracking Controller (MPTC). This generic controller, at a first stage, intends to independently fulfill several subtask objectives. These primarily independent subtask controllers are then merged into an overall controller. The controller design and analysis is based upon Lyapunov theory, which facilitates statements about stability and passivity. One of the major contributions of this paper is the design of an optimization weighting matrix that, for fully actuated robots, guarantees the passivity of a complete set of conflicting subtasks. The proposed control framework was validated in several simulations and experiments for fixed-based and free-floating robots.

In our future research, we intend to extensively compare the control concepts of Inverse Dynamics (ID), the proposed Modular Passive Tracking Control (MPTC) and other passivity-based controllers, such as the PD+ controller. This extensive comparison will be based on theoretical analysis, simulations and hardware experiments.

[15]Note: due to the modularity of MPTC, ID-based and MPTC-based tasks may be combined in the same overall control setup.

References

[1] J. Ackermann. Parameter space design of robust control systems. *IEEE Transactions on Automatic Control*, 25 (6):1058–1072, December 1980. ISSN 2334-3303. doi: 10.1109/TAC.1980.1102505.

[2] G. Antonelli. Stability Analysis for Prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5):985–994, 2009.

[3] K. Bouyarmane and A. Kheddar. On Weight-Prioritized multitask control of humanoid robots. *IEEE Transactions on Automatic Control*, 63(6):1632–1647, June 2018. ISSN 2334-3303. doi: 10.1109/TAC.2017.2752085.

[4] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, June 1997. ISSN 2374-958X. doi: 10.1109/70.585902.

[5] A. Dietrich. *Whole-Body Impedance Control of Wheeled Humanoid Robots*, volume 116. Springer International Publishing, 2016. ISBN 978-3-319-40557-5.

[6] A. Dietrich and C. Ott. Hierarchical Impedance-Based tracking control of kinematically redundant robots. *IEEE Transactions on Robotics*, 36(1):204–221, 2020. ISSN 1941-0468. doi: 10.1109/TRO.2019.2945876.

[7] A. Dietrich, C. Ott, and A. Albu-Schäffer. Multi-objective compliance control of redundant manipulators: hierarchy, control, and stability. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3043–3050, Nov 2013. doi: 10.1109/IROS.2013.6696787.

[8] J. Englsberger, A. Werner, C. Ott, B. Henze, M. A. Roa, G. Garofalo, R. Burger, A. Beyer, O. Eiberger, K. Schmid, and A. Albu-Schäffer. Overview of the torque-controlled humanoid robot TORO. In *IEEE-RAS Int. Conf. on Humanoid Robots*, pages 916–923, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041473.

[9] J. Englsberger, C. Ott, and A. Albu-Schäffer. Three-Dimensional Bipedal Walking Control Based on divergent component of motion. *IEEE Transactions on Robotics*, 31(2):355–368, April 2015. ISSN 1941-0468. doi: 10.1109/TRO.2015.2405592.

[10] J. Englsberger, G. Mesesan, A. Werner, and C. Ott. Torque-Based Dynamic Walking - a long way from simulation to experiment. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 440–447, 2018.

[11] Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014. doi: 10.1177/0278364914521306.

[12] B. Henze, M. A. Roa, and Ch. Ott. Passivity-based whole-body balancing for torque-controlled humanoid robots in multi-contact scenarios. *The International Journal of Robotics Research*, 35(12):1522–1543, 2016. doi: 10.1177/0278364916653815.

[13] M. A. Hopkins, D. W. Hong, and A. Leonessa. Compliant locomotion using whole-body control and divergent component of motion tracking. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 5726–5733, May 2015. doi: 10.1109/ICRA.2015.7140001.

[14] F. Kanehiro, H. Hirukawa, and S. Kajita. Openhrp: Open Architecture Humanoid Robotics Platform. *The International Journal of Robotics Research*, 23(2):155–165, 2004. doi: 10.1177/0278364904041324.

[15] T . Koolen, S. Bertrand, G. Thomas, T. de Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt. Design of a Momentum-Based Control Framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(01):1650007, 2016. doi: 10.1142/S0219843616500079.

[16] G. Mesesan, J. Englsberger, G. Garofalo, C. Ott, and A. Albu-Schäffer. Dynamic walking on compliant and uneven terrain using dcm and passivity-based whole-body control. In *IEEE-RAS 19th Int. Conf. on Humanoid Robots (Humanoids)*, pages 25–32, 2019.

[17] R. M. Murray. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[18] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2): 3–15, 1987. doi: 10.1177/027836498700600201.

[19] B. Paden and B. Riedle. A Positive-Real Modification of a class of nonlinear controllers for robot manipulators. In *1988 American Control Conference*, pages 1782–1785, June 1988. doi: 10.23919/ACC.1988.4790015.

[20] J. Peters, M. Mistry, F. Udwadia, J. Nakanishi, and S. Schaal. A unifying framework for robot control with redundant dofs. *Autonomous Robots*, 24:1–12, 2008. doi: 10.1007/s10514-007-9051-x.

[21] L. Righetti, J. Buchli, M. Mistry, and S. Schaal. Inverse dynamics control of floating-base robots with external constraints: A unified view. In *IEEE Int. Conf. on Robotics and Automation*, pages 1085–1090, May 2011. doi: 10.1109/ICRA.2011.5980156.

[22] L. Sentis and O. Khatib. Synthesis of Whole-Body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4): 505–518, 2005.

[23] L. Sentis, J. Park, and O. Khatib. Compliant Control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on Robotics*, 26(3):483–501, 2010.