First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback

Ewen Louis Dantec, Michel Taix, Nicolas Mansard

First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback

Ewen Dantec [a,b,*], Michel Taïx [a], Nicolas Mansard [a,b]

***Abstract*— The lack of computational power on mobile robots is a well-known challenge when it comes to implementing a real-time MPC scheme to perform complex motions. Currently the best solvers are barely able to reach 100Hz for computing the control of a whole-body legged model, while modern robots are expecting new torque references in less than 1ms. This problem is usually tackled by using a handcrafted low-level tracking control whose inputs are the low-frequency trajectory computed by the MPC. We show that a linear state feedback controller naturally arises from the optimal control formulation and can be used directly in the low-level control loop along with other sensitivities of relevant time-varying parameters of the problem. When the optimal control problem is solved by DDP, this linear controller can be computed for cheap as a by-product of the backward pass, and corresponds in part to the classical Riccati gains. A side effect of our proposition is to show that Riccati gains are valuable assets that must be used to achieve an efficient control and that they are not stiffer than the optimal control scheme itself. We propose a complete implementation of this idea on a full-scale humanoid robot and demonstrate its importance with real experiments on the robot Talos.**

I. Introduction

Model-predictive control (MPC) amounts to frequently optimizing a prediction of the robot motion and executing the beginning of the resulting trajectory while waiting for a prediction update [1]. It has lately become popular in robotics to control complex multi-variable systems such as legged or aerial robots. MPC offers a viable method to solve large-scale optimal control problem (OCP) over a receding finite horizon, while efficiently dealing with constraints, non-linearity, model uncertainties and unforeseen disturbances [2]. Over the time, the ever increasing computational power of our computers has allowed the MPC paradigm to be applied to complex systems such as 4-legged robots [3], collaborative manipulation involving dynamic obstacle [4] or quadrotor control [5].

MPC belongs to the class of trajectory optimization methods, where the trajectory planning problem is interpreted as an OCP. This approach allows to optimize simultaneously a task-related path and the associated control policy, while taking into account constraints and limitations [6]. The tasks are classically described by a non-linear cost function which enforces constraints and defines the objectives hierarchy through cost weights. At each control cycle, starting from a state estimate of the system, an OCP is solved over a receding finite-time horizon and the first elements of the resulting trajectories are forwarded to a low-level execution process. The problem is then shifted from one sampling period before being solved again. Although this method works well for linear problems, a series of challenges arises when considering non-linear systems and complex tasks such as obstacle avoidance or contact planning. Since complexity varies with the cube of the system dimension, finding a global minimum in real time to a large-scale non-linear problem is very difficult to achieve [7]. For these reasons, the use of MPC in robotics has been, until recently, limited to systems with reduced dynamics [8] or small scale [9].

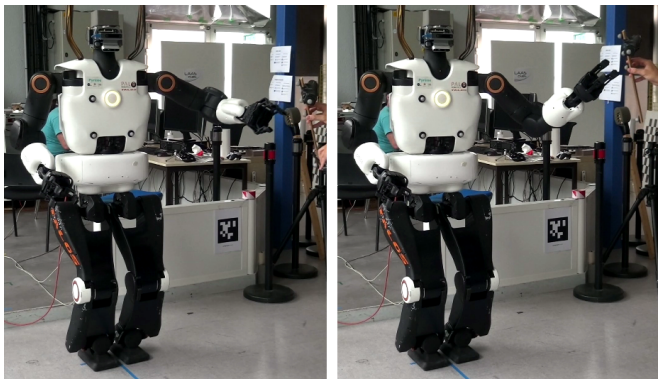

Fig. 1. The full scale humanoid Talos, used in the experiments, expects new torque references every 0.5ms. While such high frequencies is needed to unlock torque control, current MPC solvers cannot yet solve a whole-body problem at such a frequency. Another feedback must then be implemented, here to handle measurements of the robot state and target location.

Among the various numerical optimization schemes used to implement MPC, direct shooting [10] is a popular class of methods used in robotics. One of the most famous shooting algorithms is Differential Dynamic Programming (DDP) [11], [12], an efficient method featuring superlinear convergence rate with linear complexity in the horizon length, hence its success in robotics [13], [14], [15]. While it is originally formulated as a single shooting algorithm, it can be turned into a multiple shooting one [16], [10] at no extra cost [17], as we will recall later. This makes it easier to warm start the solver, either using the solution found at the previous control cycle [18], or deduced from an offline database of optimal solutions [19].

In order to tackle brutal disturbances and model uncertainties, it seems desirable to run a whole-body MPC at the highest possible frequency [20], [21], ideally around 1kHz. Direct MPC feedback at slower frequency is possible but drastically reduces the performances of the motions [22]. Moreover, most whole-body control schemes for legged

[a] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[b] Artificial and Natural Intelligence Toulouse Institute, France
[*] corresponding author: edantec@laas.fr

This work is supported by the European project MEMMO (GA-780684) and ANITI (ANR-19-P3IA-0004). We thank Guilhem Saurel (LAAS-CNRS) and Pierre Fernbach (Toward) for their experimental support.

robots work with frequencies of 400 Hz or higher [23], [24], [25], and slower frequencies are expected to decrease performances [21], [26]. For large-scale systems such as humanoid robots, recomputing the MPC at such rate may not be achievable despite some on-going exploratory works, e.g. contact-aware Linear Quadratic Regulator (LQR) controllers [27], parallelized LQR resolution [28] or multi-core GPU optimization solver [29]. Other works propose to use feedback MPC to provide an optimal feedback policy at a higher update rate than the re-computation of the whole problem. Those policies are either computed separately using another optimal control transcription [30] or based on an efficient Newton-type scheme [16]. In [31], a MPC scheme is deployed on a reduced model to rapidly generate a CoM trajectory and contact sequence, which are provided to a whole-body control block based on inverse dynamics. In [32], the feedback gains satisfy by design the contact constraints and dynamics constraints, and efficiently close the gap between the low-frequency high-level MPC and the low-level controller. However, both methods need to rely on a dedicated controller to produce a torque policy based on a state trajectory, bringing on an additional layer of complexity which should be fine-tuned.

It is known that the control solution of a constraint LQR is a continuous piece-wise affine function of the initial state [33]. Following this observation, we propose to deduce the optimal feedback policy from the OCP formulation and we show the connection between this policy and the DDP-induced Riccati gains. It is commonly acknowledged that those gains can be interpreted as the sensitivities of the optimal solution with respect to the initial state of the OCP. Sensitivities have already been used to improve the warm-start in a receding horizon scheme [16], for bilevel optimization [34] and contact simulation [35]. Differentiating an MPC has also been proposed in the purpose of adding it as a layer of a neural network [36], [37]. Because we are only interested in getting the derivative of the initial control with respect to any external parameter, our proposed derivation leads to a more efficient backward pass to compute the corresponding gain, and elegantly connects it with the Riccati matrix. This is necessary as we are proposing to use it for high-frequency control. For small state changes, this is equivalent to computing a new solution of the MPC at a much higher frequency, reaching the expected computation of the low-level control. The same approach can be generalized to any parameter of the OCP, leading to a set of Riccati-like feedback gains which can be computed at low cost by extending the classical DDP backward pass. To illustrate our method, we performed a reactive reaching task involving 28 degrees of freedom, contacts and balance over an update frequency of ~ 60 Hz for the MPC and the Riccati gains, while the low-level control is working at 2 kHz. We provide a demonstration of our whole-body MPC on the torque-controlled humanoid robot TALOS [38].

The paper is structured as follows. Sec. II introduces the basic formulation of optimal control for whole body motion in contact. Sec. III proves that Riccati gains can be interpreted as sensitivity of the problem with respect to the initial state, and gives a generic method to extract other sensitivities. Sec. IV explains the implementation of these gains in the low-level control of our scheme. Sec. V presents the experimental results on the humanoid robot TALOS.

II. Whole body optimal control with contact

A. Contact-Constrained Rigid body dynamics

Similarly to [39], we consider a rigid body system with n_j joints and n_p rigid contacts with the environment and model the dynamics of this constrained multi-body system through the following equation [40]:

$$\begin{bmatrix} \boldsymbol{M} & \boldsymbol{J}_c^\top \\ \boldsymbol{J}_c & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{q}} \\ -\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} S^\top \boldsymbol{\tau} - \boldsymbol{b} \\ -\dot{\boldsymbol{J}}_c \dot{\boldsymbol{q}} \end{bmatrix} \tag{1}$$

where $\boldsymbol{M}$ is the joint-space inertia matrix, $\boldsymbol{b}$ the generalized non-linear forces, $\boldsymbol{q} \in SE(3) \times \mathbb{R}^{n_j}$ the configuration vector which accounts for the n_j actuated joints positions and the free-flyer joint, $\dot{\boldsymbol{q}}$ the velocity vector laying in the tangent space of $SE(3) \times \mathbb{R}^{n_j}$, $\ddot{\boldsymbol{q}}$ the acceleration vector, S the actuation matrix typically selecting the actuated joints, $\boldsymbol{\tau}$ the joint torques, $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1 \cdots \boldsymbol{\lambda}_{n_p})$ and $\boldsymbol{J}_c = (\boldsymbol{J}_1 \cdots \boldsymbol{J}_{n_p})$ the concatenation of vectors of contact forces and concatenation of contact Jacobian matrices. In this context, forces $\boldsymbol{\lambda}$ abstractly represents either 3D forces for punctual contacts or spatial 6D forces for planar contacts, expressed in their respective contact frame. They have to respect the contact model described by the cone K_p:

$$\forall p = 0..n_p,\ \boldsymbol{\lambda_p} \in K_p. \tag{2}$$

We then classically consider the state of the robot to be $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{n_x}$, $\dot{\boldsymbol{x}} = (\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}) \in \mathbb{R}^{n_{dx}}$, and the control to be $\boldsymbol{u} = \boldsymbol{\tau} \in \mathbb{R}^{n_u}$. In [14], we showed that with this formulation, forces become consequences of the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the control $\boldsymbol{u}$, and thus our dynamics becomes independent from these variables. Consequently, (1) leads to a force-free partial derivative equation, where contact phases are fixed and imposed in (1), which we numerically integrate into a next-state equation:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \tag{3}$$

B. Optimal control for legged robots

Based on (3), we can formulate a continuous OCP then transcript it into a nonlinear programm in order to be solved. We use a multiple-shooting formulation, as it combines the efficiency of shooting formulations [13] with numerical stability [10]. Our discretized OCP then takes the following form:

$$\begin{aligned} \min_{\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}} & \sum_{t=0}^{T-1} \ell(\boldsymbol{x}_t, \boldsymbol{u}_t, t) + \ell_T(\boldsymbol{x}_T) \\ s.t. \quad & \boldsymbol{x}_0 = \boldsymbol{f}_0 \\ \forall t = 0..T, \quad & \boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) \end{aligned} \tag{4}$$

where the decision variables are the state $\underline{\boldsymbol{x}} = (\boldsymbol{x}_t)_{t=0..T}$ and control $\underline{\boldsymbol{u}} = (\boldsymbol{u}_t)_{t=0..T-1}$ trajectories, T is the number of knots, ℓ and ℓ_T are the running and terminal cost functions,

used to define the tasks the robot has to perform (goal tracking, center of mass tracking, etc.) and to regularize state and control trajectories, and $\boldsymbol{f}_0$ is the initial state. An OCP for locomotion often contains admissibility constraints on state and control; however, as recent progresses in handling hard constraints for large OCP [41], [42], [43] still suffer from practical limitations in the perspective of real robot experiments, we focus here on constraint-free OCP (without loss of generality for our theoretical developments [36], but targeting our experimental setup). Thus, we formulate these constraints as penalties of various forms in our cost function, and describe them in Section IV.

C. Formulating the LQR

Linearizing (4) around an initial trajectory $(\underline{\boldsymbol{x}}, \underline{\boldsymbol{u}})$ results in a LQR. Let us note $\ell(\underline{\boldsymbol{\Delta x}}, \underline{\boldsymbol{\Delta u}})$ the trajectory cost:

$$\ell(\underline{\boldsymbol{\Delta x}}, \underline{\boldsymbol{\Delta u}}) = \sum_{t=0}^{T-1} \Big(\frac{1}{2} \begin{bmatrix} \boldsymbol{\Delta x}_t^\top, \boldsymbol{\Delta u}_t^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{L_{xx}} & \boldsymbol{L_{xu}} \\ \boldsymbol{L_{ux}} & \boldsymbol{L_{uu}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta x}_t \\ \boldsymbol{\Delta u}_t \end{bmatrix} + \begin{bmatrix} \boldsymbol{\ell_x} & \boldsymbol{\ell_u} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta x}_t \\ \boldsymbol{\Delta u}_t \end{bmatrix} \Big) + \frac{1}{2} \boldsymbol{\Delta x}_T^\top \boldsymbol{L_{xx}} \boldsymbol{\Delta x}_T + \boldsymbol{\ell_x} \boldsymbol{\Delta x}_T \tag{5}$$

The LQR then writes:

$$\begin{aligned} \min_{\boldsymbol{\Delta x}, \boldsymbol{\Delta u}} \quad & \ell(\underline{\boldsymbol{\Delta x}}, \underline{\boldsymbol{\Delta u}}) \\ s.t. \quad & \boldsymbol{\Delta x}_0 = \boldsymbol{f}_0 \\ & \forall t = \{0, \cdots, T-1\}, \\ & \boldsymbol{\Delta x}_{t+1} = \boldsymbol{F_x} \boldsymbol{\Delta x}_t + \boldsymbol{F_u} \boldsymbol{\Delta u}_t + \boldsymbol{f}_{t+1} \end{aligned} \tag{6}$$

For the sake of readability, we will denote $A_b = \frac{\partial A}{\partial b}$, except when not suitable (e.g. $\frac{\partial \Delta u}{\partial x}$). Bold capital letters indicate matrices and bold letter indicate vectors. In (5), $\boldsymbol{\ell_x}, \boldsymbol{\ell_u}$ and $\boldsymbol{L_{xx}}, \boldsymbol{L_{xu}}, \boldsymbol{L_{uu}}$ are the gradients and Hessians of the cost function (indices t have been dropped to simplify derivative notations). Similarly, $\boldsymbol{F_x}$ and $\boldsymbol{F_u}$ are the Jacobians of the dynamics computed at each $(\boldsymbol{x}_t, \boldsymbol{u}_t)$. We write $\boldsymbol{f}_t$ the drift in dynamics, which represents the change in state when the control is 0. Note that $\boldsymbol{f}_t$ is typically omitted in DDP, but is introduced here to handle our multiple shooting transcription.

The optimum of (6) is characterized by the gradients of the associated Lagrangian vanishing. Let us pose $\boldsymbol{\Delta z} = (\underline{\boldsymbol{\Delta x}}, \underline{\boldsymbol{\Delta u}})$ and write the Lagrangian as:

$$\mathcal{L}(\boldsymbol{\Delta z}, \boldsymbol{\mu}) = \ell(\boldsymbol{\Delta z}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{\Delta z}) \tag{7}$$

with $\boldsymbol{h}(\boldsymbol{\Delta z})$ the vector of the T+1 dynamics equality constraints given by (6). The optimality condition can now be formulated as a linear equality called Karush-Kuhn-Tucker (KKT) condition, which binds the primal variables $\boldsymbol{\Delta z}$ to the dual Lagrange multipliers $\boldsymbol{\mu}$ associated with the dynamics constraints $\boldsymbol{h}$ and drift $\boldsymbol{f} = (\boldsymbol{f}_0, .., \boldsymbol{f}_T)$. Assuming the KKT matrix to be invertible, the LQR solution writes:

$$\begin{bmatrix} \boldsymbol{\Delta z} \\ \boldsymbol{\mu} \end{bmatrix} = - \begin{bmatrix} \mathcal{L}_{\boldsymbol{zz}} & \boldsymbol{h}_{\boldsymbol{z}}^T \\ \boldsymbol{h_z} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\ell_z} \\ \boldsymbol{f} \end{bmatrix} \tag{8}$$

While we will later exploit the KKT equation to formulate our contribution, it is in general not easy to efficiently invert the KKT matrix to solve the OCP.

D. Solving the OCP using the KKT conditions

DDP solves (6) by using Bellman's principle [44], which naturally exploits the sparsity of the Markovian nature of the dynamics constraints. Eventually, each descent step can be produced by an iterative scheme which pre-computes the descent direction in a backward pass (going from N to 0), and then finds the optimal step by line search along this direction in a forward pass (going from 0 to N) [45]. In this forward pass, the optimal state and control $(\boldsymbol{\Delta x}_t, \boldsymbol{\Delta u}_t)$ and optimal dual variable $\boldsymbol{\mu}_t$ $\forall t = 0..T-1$ can be expressed as:

$$\boldsymbol{\Delta u}_t = \boldsymbol{K}_t \boldsymbol{\Delta x}_t + \boldsymbol{k}_t \tag{9a}$$
$$\boldsymbol{\mu}_{t+1} = \boldsymbol{V}_{t+1}(\boldsymbol{F}_x \boldsymbol{\Delta x}_t + \boldsymbol{F}_u \boldsymbol{\Delta u}_t) + \boldsymbol{v}_{t+1} \tag{9b}$$

Here the gains and value fonction are computed during the backward pass of the DDP: starting from $\boldsymbol{V}_T = \boldsymbol{L_{xx}}$ and $\boldsymbol{v}_T = \boldsymbol{\ell_x} + \boldsymbol{L_{xx}} \boldsymbol{f}_T$, we have, $\forall t = 0..T-1$:

$$\boldsymbol{K}_t = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1} \boldsymbol{Q_{ux}} \tag{10a}$$
$$\boldsymbol{k}_t = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1} \boldsymbol{q_u} \tag{10b}$$
$$\boldsymbol{V}_t = \boldsymbol{Q_{xx}} - \boldsymbol{Q_{xu}} \boldsymbol{K}_t \tag{10c}$$
$$\boldsymbol{v}_t = \boldsymbol{q_x} - \boldsymbol{Q_{xu}} \boldsymbol{k}_t + \boldsymbol{V}_t \boldsymbol{f}_t \tag{10d}$$

The backward pass also computes the derivatives of the quality function:

$$\boldsymbol{q_x} = \boldsymbol{\ell_x} + \boldsymbol{F}_{\boldsymbol{x}}^T \boldsymbol{v}_{t+1} \tag{11a}$$
$$\boldsymbol{q_u} = \boldsymbol{\ell_u} + \boldsymbol{F}_{\boldsymbol{u}}^T \boldsymbol{v}_{t+1} \tag{11b}$$
$$\boldsymbol{Q_{xx}} = \boldsymbol{L_{xx}} + \boldsymbol{F}_{\boldsymbol{x}}^T \boldsymbol{V}_{t+1} \boldsymbol{F_x} \tag{11c}$$
$$\boldsymbol{Q_{xu}} = \boldsymbol{L_{xu}} + \boldsymbol{F}_{\boldsymbol{x}}^T \boldsymbol{V}_{t+1} \boldsymbol{F_u} \tag{11d}$$
$$\boldsymbol{Q_{uu}} = \boldsymbol{L_{uu}} + \boldsymbol{F}_{\boldsymbol{u}}^T \boldsymbol{V}_{t+1} \boldsymbol{F_u} \tag{11e}$$

DDP indeed accounts for the second-order derivatives of the dynamics, that we ignore here for simplicity. This approach is classically described as iterative LQR (iLQR) [46]. All the following derivations, that we write without the hessian of the dynamics, directly extend to the full DDP formulation. As exploited next, DDP can be interpreted as an operator to efficiently evaluate the left multiplication by the inverse of the KKT (8).

III. Sensitivity analysis of the optimal solution

Suppose that our OCP depends on a parameter $\boldsymbol{\theta}$ which varies with time. If the variation of $\boldsymbol{\theta}$ is non-negligible over the computation duration of the OCP, then the optimal solution is no longer correct since it has been computed over an outdated value of $\boldsymbol{\theta}$. Determining the sensitivity of the OCP with respect to $\boldsymbol{\theta}$ will allow us to approximate a control correction proportional to the variation of the parameter.

A. Defining sensitivity as partial derivative of the OCP

Let us view the OCP described by (4) as a classical non-linear problem parametrized by a given $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$. We define the optimal state and control trajectory $\underline{\boldsymbol{x}}^* = (\boldsymbol{x}_t^*)_{t=0..T}$ and $\underline{\boldsymbol{u}}^* = (\boldsymbol{u}_t^*)_{t=0..T-1}$. For the sake of simplicity we denote by

$\boldsymbol{z}^* = (\underline{\boldsymbol{x}}^*, \underline{\boldsymbol{u}}^*) \in \mathbb{R}^{n_z}$ with $n_z = Tn_u + (T+1)n_x$ solution of the following discretized problem:

$$\begin{aligned} \mathcal{P}(\boldsymbol{\theta}) = \boldsymbol{z}^* = \underset{\boldsymbol{z}}{\text{argmin}} \;\; & \ell(\boldsymbol{z}, \boldsymbol{\theta}) \\ s.t. \qquad & \boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta}) = 0 \end{aligned} \tag{12}$$

with $\boldsymbol{h} : \mathbb{R}^{n_z} \times \mathbb{R}^{n_\theta} \longrightarrow \mathbb{R}^{(T+1)\times n_x}$ the $T+1$ discretized dynamics constraints given by $\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t)\ \forall t = 0..T-1$ and $\boldsymbol{x}_0 = \boldsymbol{f}_0$. We assume ℓ and $\boldsymbol{h}$ are continuously twice differentiable, and that the Mangasarian-Fromovitz constraint qualification holds for our nonlinear problem, in order to be able to apply the KKT optimality conditions. From here the Lagrangian of the problem writes $\mathcal{L}(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\theta}) = \ell(\boldsymbol{z}, \boldsymbol{\theta}) + \boldsymbol{\mu}^T \boldsymbol{h}(\boldsymbol{z}, \boldsymbol{\theta})$ and the KKT optimality conditions state that if $\boldsymbol{z}^*$ is solution of the problem (12) for $\boldsymbol{\theta}$, then:

$$\begin{aligned} \mathcal{L}_{\boldsymbol{z}}(\boldsymbol{z}^*, \boldsymbol{\mu}, \boldsymbol{\theta}) = \ell_{\boldsymbol{z}}(\boldsymbol{z}^*, \boldsymbol{\theta}) + \boldsymbol{\mu}^T \boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}^*, \boldsymbol{\theta}) = 0 \\ \boldsymbol{h}(\boldsymbol{z}^*, \boldsymbol{\theta}) = 0 \end{aligned} \tag{13}$$

Finally we suppose that for each $\boldsymbol{\mu} \in \mathbb{R}^{(T+1)\cdot n_x}$ satisfying the KKT conditions and each $\boldsymbol{\Delta z} \neq 0 \in \mathbb{R}^{n_z}$ so that $\boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}^*, \boldsymbol{\theta})\boldsymbol{\Delta z} = 0$, we have the following inequality:

$$\boldsymbol{\Delta z}^T \mathcal{L}_{\boldsymbol{zz}}(\boldsymbol{z}^*, \boldsymbol{\mu}, \boldsymbol{\theta}) \boldsymbol{\Delta z} > 0 \tag{14}$$

Under these assumptions, Shapiro [47] showed that the function $\mathcal{P}(\cdot)$ is directionally differentiable and that for each $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, $\boldsymbol{\Delta\theta} \in \mathbb{R}^{n_\theta}$, there exists $\boldsymbol{\mu}$ verifying the KKT conditions such that $\mathcal{P}'(\boldsymbol{\theta}; \boldsymbol{\Delta\theta})$, derivative along the direction $\boldsymbol{\Delta\theta}$, is the unique solution of the following quadratic program:

$$\begin{aligned} \underset{\boldsymbol{\Delta z}}{\text{argmin}} \; & \frac{1}{2} \boldsymbol{\Delta z}^T \mathcal{L}_{\boldsymbol{zz}}(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\theta}) \boldsymbol{\Delta z} + \boldsymbol{\Delta z}^T \mathcal{L}_{\boldsymbol{z\theta}}(\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\theta}) \boldsymbol{\Delta\theta} \\ s.t. \qquad & \boldsymbol{h}_{\boldsymbol{z}}(\boldsymbol{z}, \boldsymbol{\theta}) \boldsymbol{\Delta z} = 0 \end{aligned} \tag{15}$$

Since in our case, the Lagrangian multiplier $\boldsymbol{\mu}$ is unique due to the upper diagonal form of the constraints matrix, it turns out we can easily compute the directional derivative of our problem along the direction $\boldsymbol{\Delta\theta}$, and from it immediately deduce the sensitivities of (12) under a KKT-like shape:

$$\begin{bmatrix} \frac{\partial \boldsymbol{\Delta z}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}} \end{bmatrix} = - \begin{bmatrix} \mathcal{L}_{\boldsymbol{zz}} & \boldsymbol{h}_{\boldsymbol{z}}^T \\ \boldsymbol{h}_{\boldsymbol{z}} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathcal{L}_{\boldsymbol{z\theta}} \\ 0 \end{bmatrix} \tag{16}$$

In this equation, $\frac{\partial \boldsymbol{\mu}}{\partial \theta}$ is the derivative of a Lagrangian multiplier increment we are not interested in. On the other hand, $\frac{\partial \boldsymbol{\Delta z}}{\partial \theta} \Delta\theta$ is the correction to apply to the state and control trajectories when the parameter error is $\boldsymbol{\Delta\theta}$. As literally solving (16) would be inefficient to find the derivatives, we now show how to exploit DDP to evaluate the sensitivities.

B. Computing the sensitivity through the backward pass

Let us go back to the linearized problem described by (6). The matrix in (16) is the inverse of the KKT matrix already met in (8), yet not multiplied here by $(-\boldsymbol{\ell}_{\boldsymbol{z}}, -\boldsymbol{f})$ but by $(-\mathcal{L}_{\boldsymbol{z\theta}}, 0)$. As explained in Sec. II-D, DDP can be interpreted as an efficient operator evaluating the multiplication by the inverse of the KKT matrix. We then propagate DDP backward and forward recurrences in (16) by replacing the terms $(\boldsymbol{\ell}_{\boldsymbol{x}}, \boldsymbol{\ell}_{\boldsymbol{u}}, \boldsymbol{f}_t)$ of (10) and (11) with $(\mathcal{L}_{\boldsymbol{x\theta}}, \mathcal{L}_{\boldsymbol{u\theta}}, 0)$. This immediately leads to the following backward pass:

$$\boldsymbol{K}_{\boldsymbol{\theta}} = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1} \boldsymbol{Q}_{\boldsymbol{u\theta}} \tag{17a}$$
$$\boldsymbol{V}_{\boldsymbol{\theta}} = \mathcal{L}_{\boldsymbol{x\theta}} \quad \text{for} \quad t = T \tag{17b}$$
$$\boldsymbol{V}_{\boldsymbol{\theta}} = \boldsymbol{Q}_{\boldsymbol{x\theta}} - \boldsymbol{Q}_{\boldsymbol{xu}} \boldsymbol{K}_{\boldsymbol{\theta}} \tag{17c}$$
$$\boldsymbol{Q}_{\boldsymbol{x\theta}} = \mathcal{L}_{\boldsymbol{x\theta}} + \boldsymbol{F}_{\boldsymbol{x}}^T \boldsymbol{V}'_{\boldsymbol{\theta}} \tag{17d}$$
$$\boldsymbol{Q}_{\boldsymbol{u\theta}} = \mathcal{L}_{\boldsymbol{u\theta}} + \boldsymbol{F}_{\boldsymbol{u}}^T \boldsymbol{V}'_{\boldsymbol{\theta}} \tag{17e}$$

where (17a) reflects (10a), (17c) reflects (10c), and (17d)-(17e) reflect (11a)-(11b). As before, t is dropped in index when it conflicts with another notation to improve readability. The value derivative $\boldsymbol{V}_\theta$ at time $t+1$ is denoted $\boldsymbol{V}'_\theta$. Similarly propagating the forward pass on (16) gives:

$$\begin{aligned} \frac{\partial \boldsymbol{\Delta u}_t}{\partial \boldsymbol{\theta}} &= \boldsymbol{K}_t \frac{\partial \boldsymbol{\Delta x}_t}{\partial \boldsymbol{\theta}} + \boldsymbol{K}_{\boldsymbol{\theta}} \\ \frac{\partial \boldsymbol{\Delta x}_{t+1}}{\partial \boldsymbol{\theta}} &= \boldsymbol{F}_x \frac{\partial \boldsymbol{\Delta x}_t}{\partial \boldsymbol{\theta}} + \boldsymbol{F}_u \frac{\partial \boldsymbol{\Delta u}_t}{\partial \boldsymbol{\theta}} \end{aligned} \tag{18}$$

In what follows, we are mostly interested by $\frac{\partial u_0}{\partial \boldsymbol{\theta}}$, sensitivity of the first control with respect to the parameter $\boldsymbol{\theta}$. Since the OCP (4) has the same derivatives as the LQR (6) at each knot, the sensitivity of both problems are equal:

$$\frac{\partial \boldsymbol{u}_0^*}{\partial \boldsymbol{\theta}} = \frac{\partial \boldsymbol{\Delta u}_0^*}{\partial \boldsymbol{\theta}} \tag{19}$$

C. Interpreting Riccati gains as sensitivity

A feedback policy can be obtained from (18) when $\boldsymbol{\theta}$ is measured at a higher frequency than the DDP solves (4). In the experimental part of this paper, we demonstrate MPC feedback from state and vision, where $\boldsymbol{\theta}$ contains the robot state $\boldsymbol{x}_0$ and visual target $\boldsymbol{p}$.

Consider first the case where $\boldsymbol{\theta} = \boldsymbol{x}_0$. The only derivatives of the problem which depend on $\boldsymbol{x}_0$ are $\mathcal{L}_{x,0}$ and $\mathcal{L}_{u,0}$, so by recurrence with (17), it is straightforward to deduce that $\boldsymbol{V}_{\boldsymbol{x}_0} = 0\ \forall t = 1..T$ and that $\boldsymbol{K}_{\boldsymbol{x}_0} = -\boldsymbol{Q}_{\boldsymbol{uu}}^{-1} \mathcal{L}_{\boldsymbol{ux_0}} = 0\ \forall t = 0..T$. The derivative of the first optimal control with respect to its initial state then simply writes:

$$\frac{\partial \boldsymbol{\Delta u}_0}{\partial \boldsymbol{x}_0} = \boldsymbol{K}_0 \frac{\partial \boldsymbol{\Delta x}_0}{\partial \boldsymbol{x}_0} + \boldsymbol{K}_{\boldsymbol{x}_0} = \boldsymbol{K}_0 \tag{20}$$

This equation shows, as it is already known, that the Riccati gains $\boldsymbol{K}_0$ can be interpreted as the sensitivity to the initial state. Given that the current state of the robot is provided at a sufficiently high frequency, the Riccati gains act as a feedback term which approximates the optimal control between two LQR computations.

Let us now suppose that $\boldsymbol{\theta} = \boldsymbol{p}$, defining a visual target. Since $\frac{\partial \boldsymbol{\Delta x}_0}{\partial \boldsymbol{p}} = 0$, the derivative of the optimal control simply writes $\frac{\partial \boldsymbol{\Delta u}_0}{\partial \boldsymbol{p}} = \boldsymbol{K}_p$, with $\boldsymbol{K}_p$ depending on the cost derivatives through the back-propagation (17).

The resulting feedback policy is a first order Taylor development of the MPC: suppose that the MPC has been solved at a given observed state $(\boldsymbol{x}_0, \boldsymbol{p}_0)$, denoted by $\boldsymbol{u}_0 =$

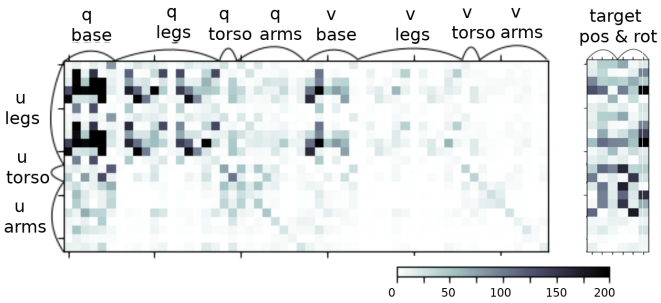

Fig. 2. Left: matrix of the absolute Riccati gain $\boldsymbol{K}_0$ for a whole-body OCP involving end-effector tracking and biped stabilization. Right: matrix of the absolute placement (position and rotation) gains for a whole-body OCP involving tracking a reference placement with the left hand while stabilizing balance.

$mpc(\boldsymbol{x}_0, \boldsymbol{p}_0)$. The linear feedback at a new observation $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}})$ will then approximate the MPC solution:

$$mpc(\hat{\boldsymbol{x}}, \hat{\boldsymbol{p}}) \approx mpc(\boldsymbol{x}_0, \boldsymbol{p}_0) + \boldsymbol{K}_0(\hat{\boldsymbol{x}} - \boldsymbol{x}_0) + \boldsymbol{K}_p(\hat{\boldsymbol{p}} - \boldsymbol{p}_0) \quad (21)$$

It has been considered that the Riccati gains might be too stiff to be actually used for feedback [48]. Yet we see here that they are nothing more than a linear interpolation of the MPC feedback, and, for a sufficiently high frequency, they lead to a fair numerical approximation of the MPC, hence are not stiffer than the MPC itself.

The structures of typical feedback gains for initial state and tracking target are presented in Fig. 2. It is interesting to note that the Riccati matrix for initial state presents a strong diagonal in position: the most contributing correction of a given joint torque strongly depends on the position error of this joint. It can also be noted that the feedback correction associated to the base position of the robot is very high, and affects mainly the leg controls. This is expected since the legs are the main drivers of the robot base position. Finally, one can notice that the velocity feedback term of the Riccati matrix are small compared to the position feedback terms, which is reassuring since velocity estimates are typically more noisy, and LQR controllers tend to create brutal velocity feedbacks [27].

The feedback gains for the target $\boldsymbol{p}$ presented in Fig. 2 mainly act on the 4 joints of the left arm, as expected since the end-effector considered in this figure is the left wrist.

IV. MPC Implementation with Riccati Feedback

A. An OCP for reaching, balancing and switching contacts

The OCP formulated for the experiments is composed of five different costs:

- a state regularization cost (weight 0.02): $\ell_1(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_d)^T \boldsymbol{R}_x (\boldsymbol{x} - \boldsymbol{x}_d)$ with R_x a positive definite weight matrix and $\boldsymbol{x}_d$ the default state;
- a control regularization cost (weight 0.001): $\ell_2(\boldsymbol{u}) = (\boldsymbol{u} - \boldsymbol{u_d})^T \boldsymbol{R}_u (\boldsymbol{u} - \boldsymbol{u_d})$ with R_u a positive definite weight matrix and $\boldsymbol{u}_d$ the gravity-compensating torque in default state;
- a goal-tracking cost (weight 15): $\ell_3(\boldsymbol{x}) = a(\boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{p}_d)$ with $a : \boldsymbol{r} \mapsto log(1 + \frac{||\boldsymbol{r}||}{\alpha})$ a logarithmic activation function with $\boldsymbol{p}(\boldsymbol{x})$ current end-effector position, $\boldsymbol{p}_d$ desired end-effector position and $\alpha = 0.2$;
- a Center of Mass (CoM) tracking cost (weight 600): $\ell_4(\boldsymbol{x}) = ||\boldsymbol{c}(\boldsymbol{x}) - \boldsymbol{c}_d||^2$ with $\boldsymbol{c}(\boldsymbol{x})$ current CoM and $\boldsymbol{c}_d$ desired CoM;
- a kinematic limit cost (weight 1000): $\ell_5(\boldsymbol{x}) = ||max(\boldsymbol{x} - \boldsymbol{x}_u, \boldsymbol{0}) + min(\boldsymbol{x} - \boldsymbol{x}_l, \boldsymbol{0})||^2$ with $\boldsymbol{x}_u$ the upper bound and $\boldsymbol{x}_l$ the lower bound of the joints positions and velocities.

For all experiments, the contact phases are imposed and the CoM position reference remains fixed in the center of the support polygon in order to prevent the CoM from drifting out of balance. The friction constraint described in (2) is then trivially respected.

The OCP is composed of 100 different knots separated by a 10ms time step. It is first solved until convergence before starting the motion; then, a single iLQR iteration is performed, using the previous solution as an initial warm-start, before iterating with the latest sensor measurements. Two quantities are estimated from sensors: the robot state $\boldsymbol{x}_0$ (from joint encoders and base IMU), and the tracking target position $\boldsymbol{p}_d$ (from motion capture camera). This formulation is sufficient to ensure stable reaching motions with both feet in contact and to reject external disturbances in real time.

B. Low-level control

The low-level torque control is composed of a proportional-derivative feedback on the joint torque measurement, plus a feedforward term which compensates for the intrinsic dynamics of the joints which are not considered in the whole-body model. Unexpected dynamics such as motor inertia, high frictions or inner flexibility of the harmonic-drive are thus handled by the low-level control. This way every joint behaves as an ideal joint from the point of view of the high-level control MPC.

C. Riccati interpolation of the MPC

Our DDP scheme produces the optimal control along with the sensitivities associated with the initial state and desired position of the end-effector. The first sensitivity $\boldsymbol{K}_0 = \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}$ has been introduced in Sec. III-C and is directly obtained during the backward pass of the DDP. The second sensitivity $\boldsymbol{K}_p = \frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{p}}$ is obtained by noting that the only cost which depends on the target position is the goal-tracking cost $\ell_3(\boldsymbol{x})$. The only non-zero derivative of the Lagrangian gradient then writes:

$$\mathcal{L}_{\boldsymbol{xp},t} = -\boldsymbol{J}_{ee}^{\top} \boldsymbol{A} \quad (22)$$

with $\boldsymbol{J}_{ee}(\boldsymbol{x})$ Jacobian of the end-effector when the state is $\boldsymbol{x}$ and $\boldsymbol{A}$ Hessian of the activation function. Re-injecting in (17), we obtained the sensitivity $\boldsymbol{K_p}$ which is represented by the first three columns of the right matrix of Fig. 2.

D. ROS architecture

The architecture is illustrated in Fig. 3, and is composed of two parallel processes running on independent CPUs: one for the MPC, the other for the low-level torque control.

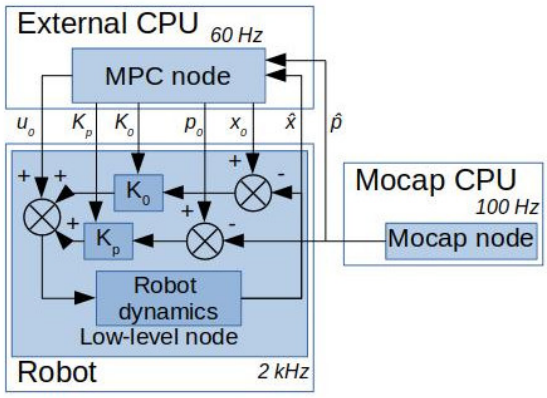

Fig. 3. Diagram of the ROS implementation of the Riccati feedback MPC.

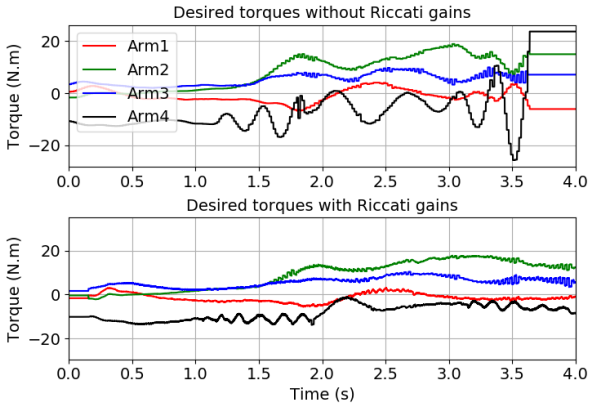

Fig. 4. Experiment A: whole-body tracking experiment: without Riccati gains, the desired torque oscillations become too large after 4 seconds, and the securities of the robot are triggered, shutting down all motors. With Riccati gains, the torque trajectory remains stable.

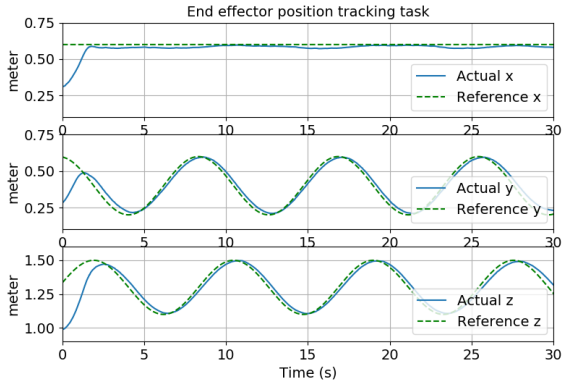

Fig. 5. Experiment A: tracking plot for the MPC scheme with Riccati gains. Tracking weights are two times higher than in Fig. 4 (30 instead of 15).

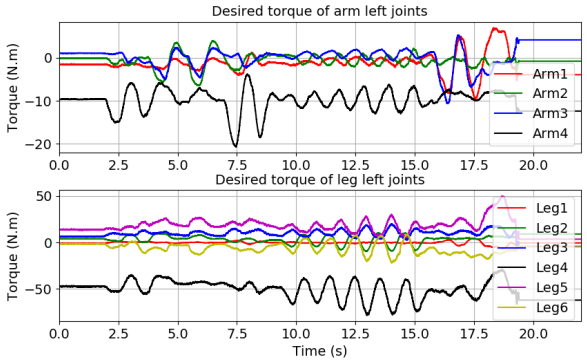

Fig. 6. Experiment B: the MPC is shut down and external disturbances are manually applied on the left arm and shoulder.

In addition, a motion capture system (mocap) measures at 100Hz the position of the target to be reached and sends it to the other two nodes. In order to synchronise the different frequencies of our processes, we re-use the same ROS publisher-subscriber architecture presented in [22].

V. Experimental Results

The experiments evaluate the interest of the linear feedback for direct state feedback using $\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{x}_0}$ (H1) and direct feedback to the target position $\boldsymbol{p}$ using $\frac{\partial \boldsymbol{u}_0}{\partial \boldsymbol{p}}$ (H2).

In order to benchmark these feedbacks, three different experimental protocols have been set up on the TALOS robot [38], in which the MPC node is running on a powerful external computer (AMD Ryzen 5950X, 32 cores and 4.9GHz with 64 Go of RAM) whereas the low-level control node is running on the robot internal computer. We used Pinocchio to enable fast computations of costs, dynamics and their derivatives [49], [50], allowing the OCP to be solved at approximately 60 Hz. Two of the experiments are based on the same whole-body tracking task, which requires the robot to reach a moving target with the end-effector, here its left hand, while standing on the ground and keeping its balance. We used 22 joints in our model (wrists and neck are kept fixed for practical reasons), plus the free flyer state which is estimated by an observer provided by the manufacturer. Our companion video, also available at *https://gepettoweb.laas.fr/articles/dantec22.html*, illustrates these experiments as well as other dynamic motions made possible by the use of Riccati interpolation.

A. Experience A: Riccati feedback experiment

To test (H1), the whole-body tracking task was performed with and without the use of Riccati gains in the low-level control loop. For this experiment, the target $\boldsymbol{p}_d$ is not estimated from sensor but set to an arbitrary sinusoidal trajectory. Fig. 4 shows that the MPC alone produces higher reaction peaks (due to delay) which will eventually trigger the robot inner securities on torque and velocity. With the Riccati gains, the oscillatory peaks remain limited. Tracking accuracy can then be safely improved by raising the weight of the tracking cost, leading to Fig. 5. Note that without Riccati gains, it is not safe to use such weight on the robot.

We also experimented with a fixed gain matrix. While our robot is torque controlled (i.e. low-level motor controllers feedback on measured torques), we have copied the PD gains typically used when the robot is position controlled. Yet the behavior in simulation remains unstable and similar to no feedback. As shown in Fig. 2, the off-diagonal Ricatti terms cannot be ignored.

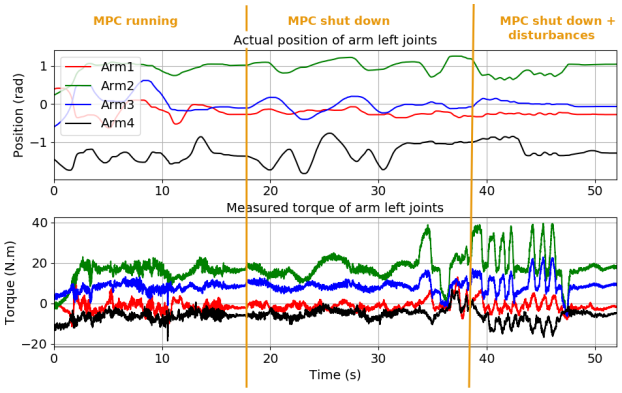

Fig. 7. Experiment C: placement feedback state and control plots: MPC is shut down at $t = 18s$, and external disturbances start to be applied at $t = 38s$.

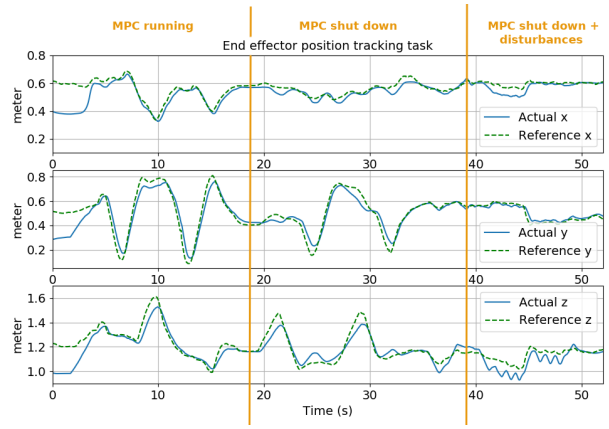

Fig. 8. Experiment C: placement feedback tracking plot.

B. Experience B: disturbances experiment

Our second experimental protocol aims at showing that direct state feedback (H1) alone produces a decent control policy able to stabilize the robot under external disturbances. The experiment consists in launching the MPC with no goal-tracking cost, then shutting it down and applying external disturbances to the robot. The policy sent to the low-level control then becomes $\boldsymbol{u}^* = \boldsymbol{u}_0 + \boldsymbol{K}_0(\hat{\boldsymbol{x}} - \boldsymbol{x}_0)$, with $\hat{\boldsymbol{x}}$ being the only varying quantity. This formulation is approximating the optimal stabilizing policy around $\boldsymbol{x}_0$. Here $(\boldsymbol{u}_0, \boldsymbol{K}_0, \boldsymbol{x}_0)$ are drawn from the last DDP computation of the high-level control. As can be seen in Fig. 6, the Riccati gains efficiently reject the perturbations and produce a smooth stabilizing torque in response to it. While our goal is not to promote a purely linear feedback, this experiment illustrates how large the stable domain of our linear policy can be.

C. Experiment C: placement feedback experiment

Our third experimental protocol aims at testing (H2) by implementing a feedback policy on the target position, as described in Sec. IV-C. This time the mocap system is used to provide the desired placement of the end-effector at a frequency of 100 Hz. Given that this frequency is similar to the MPC node frequency, the target feedback is small compared to the feedforward term $\boldsymbol{u}_0^*$. To better observe the feedback effect, the MPC has been shut down so that the drift in desired placement becomes significant over time.

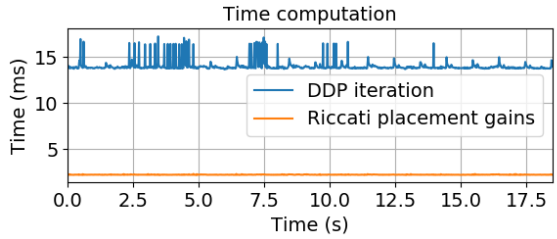

Fig. 9. Experiment C: placement feedback time plot.

The experiment is composed of three different parts: first, the MPC node is started and the OCP is solved at approximately 60 Hz. Then, at $t = 18s$, we shut down the MPC but keep the target moving to illustrate the effect of the placement feedback gains. From this moment the control is only due to state and target feedbacks. Finally, starting from $t = 38s$, external disturbances are applied to the left arm the robot. Using the sensitivities to approximate the optimal solution is relevant as long as the current OCP, defined by the current state and target position, does not vary too much from the OCP computed just before shutting down the MPC.

The resulting plots of this experiment are presented in Fig 7 and Fig. 8. Switching off the MPC and relying only on the Riccati gains lead the robot state to remain in the vicinity of the initial state used by the MPC node to compute the last OCP solution. As a consequence, the accuracy of the tracking decreases after the MPC shutdown. Nevertheless, placement gains provide a coarse approximation of the optimal control and are sufficient to move the end effector toward the desired target and reject external perturbations, as observed in Fig. 8.

Fig. 9 provides the computation cost of the DDP and the extra loop computing $\boldsymbol{K}_p$. As expected, $\boldsymbol{K}_p$ is much cheaper to compute as it always corresponds to back-propagating only 3 columns, as opposed to the classical backward pass which works with the n_x columns of $\boldsymbol{K}_t$.

VI. Conclusion

This paper highlights the idea that the classical feedback gains obtained during the Riccati recursion of the DDP can be used as sensitivities with respect to the initial state of the OCP. These gains allow to interpolate the optimal control at 2kHz in order to produce dynamical yet stable motions which would otherwise fail because of the MPC computation time delay. This has been illustrated on a whole-body tracking movement on the full scale humanoid robot Talos. Additionally, other sensitivities can be computed and used at 2kHz in order to adapt the control to a rapidly changing parameter, e.g. the desired end-effector position. This is especially useful to deduce at high frequency a good approximation of the optimal policy when small perturbations are observed by fast sensors. This work paves the way for the implementation of a real-time whole-body torque MPC with dynamic contacts, able to perform challenging tasks like walking or interacting with the environment, at arbitrary high feedback frequency. Theoretical extensions to hand constraints and contact-invariant formulation and experimental extensions to dynamic locomotion will be considered in the future.

REFERENCES

[1] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation and Design*. Nob Hill Publishing, 2017.

[2] T. A. Johansen and S. Skogesta, *Selected topics on constrained and nonlinear control*, M. Huba *et al.*, Eds. STU-NTNU, 2011, (Chapter 1.).

[3] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," *IEEE-RAS Humanoids*, 2017.

[4] M. Krämer, C. Rösmann, F. Hoffmann, and T. Bertram, "Model predictive control of a collaborative manipulator considering dynamic obstacles," *Optim Control Appl Meth. 2020;41:1211–1232.*, 2020.

[5] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control," *Mediterranean Conference on Control Automation (MED)*, 2011.

[6] R. Findeisen and F. Allgöwer, "An introduction to nonlinear MPC," *Benelux Meeting on Systems and Control*, 2020.

[7] S. J. Qin and T. A. Badgwell, "An overview of nonlinear model predictive control applications," *Nonlinear Predictive Control, Springer–Verlag, pp 369—-392*, 2000.

[8] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," *IEEE ICRA*, 2003.

[9] M. Neunert, C. De Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli, "Fast nonlinear model predictive control for unified trajectory optimization and tracking," in *IEEE ICRA*, 2016.

[10] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[11] D. Q. Mayne, "Differential dynamic programming–a unified approach to the optimization of dynamic systems," *Control and Dynamics Systems*, vol. 10, pp. 179–254, 1973.

[12] Y. Tassa, T. Erez, and W. D. Smart, "Receding horizon differential dynamic programming," *Advances in Neural Information Processing Systems*, vol. 20, p. 1465–1472, 2008.

[13] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ IROS*, 2012.

[14] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "DDP for multi-phase rigid contact dynamics," *IEEE-RAS Humanoids*, 2018.

[15] B. Jackson, Z. Manchester, and F. Farshidian, "Software tools for real-time optimal control," *Workshop at RSS*, 2021.

[16] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on control and optimization*, 2005.

[17] C. Mastalli, R. Budhiraja *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," *IEEE ICRA*, 2020.

[18] M. Diehl, R. Findeisen *et al.*, "An efficient algorithm for nonlinear model predictive control of large-scale systems. Part I: Description of the method." *Automatisierungstechnik*, pp. 557–567, 2002.

[19] N. Mansard, A. Del Prete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," *IEEE ICRA*, 2018.

[20] B. Henze, A. Werner, M. A. Roa, G. Garofalo, J. Englsberger, and C. Ott, "Control applications of toro—a torque controlled humanoid robot," in *IEEE-RAS Int. Conf. Humanoid Robots*, 2014.

[21] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *IEEE/RSJ IROS*, 2014.

[22] E. Dantec, R. Budhiraja *et al.*, "Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos," *IEEE-ICRA*, 2021.

[23] O. Melon, R. Orsolino *et al.*, "Receding-horizon perceptive trajectory optimization for dynamic legged locomotion with learned initialization," *IEEE-ICRA*, 2021.

[24] G. Raiola, E. M. Hoffman, M. Focchi, N. Tsagarakis, and C. Semini, "A simple yet effective whole-body locomotion framework for quadruped robots," *Front. Robot. AI 7:528473. doi: 10.3389/frobt.2020.528473*, 2020.

[25] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse dynamics control," *International Journal of Robotics Research*, vol. 32, 03 2013.

[26] A. D. Prete, "Control of contact forces using whole-body force and tactile sensors: Theory and implementation on the icub humanoid robot," Ph.D. dissertation, Italian Institute of Technology and University of Genova, 2013.

[27] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and walking using full dynamics lqr control with contact constraints," *IEEE-RAS Humanoids*, 2016.

[28] F. Laine and C. Tomlin, "Parallelizing lqr computation through endpoint-explicit riccati recursion," *IEEE CDC*, 2019.

[29] B. Chrétien, A. Escande, and A. Kheddar, "Gpu robot motion planning using semi-infinite nonlinear programming," *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[30] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," *IEEE-ICRA*, 2016.

[31] C. Mastalli, I. Havoutis, M. Focchi, D. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping and whole-body control," *IEEE Transactions on Robotics*, 2020.

[32] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," *IEEE/RSJ IROS*, 2019.

[33] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica 38(1):3-20*, 2002.

[34] W. Chen, Z. Shao, and L. Biegler, "A bilevel nlp sensitivity-based decomposition for dynamic optimization with moving finite elements," *AIChE Journal*, vol. 60, 2014.

[35] Q. L. Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 3413-3420*, 2021.

[36] B. Amos, I. Rodriguez, J. Sacks, B. Boots, and Z. Kolter, "Differentiable mpc for end-to-end planning and control," *NeurIPS*, 2018.

[37] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," *ICML*, 2017.

[38] O. Stasse, T. Flayols *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," *IEEE-RAS Humanoids*, 2017.

[39] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2014.

[40] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and sparse resolution of constrained dynamic equations," in *R:SS*, 2021.

[41] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," *IEEE/RSJ IROS*, 2019.

[42] S. Kazdadi, J. Carpentier, and J. Ponce, "Equality constrained differential dynamic programming," *IEEE ICRA*, 2021.

[43] R. Verschueren, G. Frison, D. Kouzoupis, N. van Duijkeren, A. Zanelli, B. Novoselnik, J. Frey, T. Albin, R. Quirynen, and M. Diehl, "Acados: a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, pp. 1–37, 2021.

[44] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[45] Y. Tassa, "Theory and implementation of biomimetic motor controllers," Ph.D. dissertation, Hebrew University of Jerusalem, 2011.

[46] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," *Int.Conf. on Informatics in Control, Automation and Robotics*, 2004.

[47] A. Shapiro, "Sensitivity analysis of nonlinear programs and differentiability properties of metric projections," *SIAM Journal on Control and Optimization 26*, 1988.

[48] S. Mason, N. Rotella, S. Schaal, and L. Righetti, "Balancing and walking using full dynamics lqr control with contact constraints," in *IEEE-RAS Humanoids*, 2016.

[49] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and Systems*, 2018.

[50] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," *IEEE/SICE Int. Symp. on System Integration*, 2019.