High-Frequency Nonlinear Model Predictive Control of a Manipulator

Sébastien Kleff, Avadesh Meduri, Rohan Budhiraja, Nicolas Mansard, Ludovic Righetti

High-Frequency Nonlinear Model Predictive Control of a Manipulator

Sébastien Kleff[1,2], Avadesh Meduri[1], Rohan Budhiraja[2], Nicolas Mansard[2,4], Ludovic Righetti[1,3]

***Abstract*— Model Predictive Control (MPC) promises to endow robots with enough reactivity to perform complex tasks in dynamic environments by frequently updating their motion plan based on measurements. Despite its appeal, it has seldom been deployed on real machines because of scaling constraints. This paper presents the first hardware implementation of closed-loop nonlinear MPC on a 7-DoF torque-controlled robot. Our controller leverages a state-of-the art optimal control solver, namely Differential Dynamic Programming (DDP), in order to replan state and control trajectories at real-time rates ($1kHz$). In addition to this experimental proof of concept, we present exhaustive performance analysis on the iconic pick-and-place task and show that our controller outperforms open-loop MPC. We also exhibit the importance of a sufficient preview horizon and full robot dynamics in the controller performance through comparisons with inverse dynamics and kinematic optimization.**

I. Introduction

In order for robots to perform complex tasks such as collaborating with a human or running out in the real world, they must be able to anticipate and react. Optimal control offers a convenient framework to endow them with such capabilities as it characterizes state-dependent control policies from high-level objectives over a time horizon. Model predictive control (MPC) [1], [2] approximates the policy online into locally optimal policies. Although well-known within the robotics community [1], [3], its hardware deployment is yet to be achieved when using high dimensional nonlinear dynamic models, mainly due to scaling issues. In default of sufficient computation capabilities two main approaches can be found in the literature.

One approach is to separate planning and control into offline or low-frequency motion planning and online inverse dynamics (ID) [4]. Modern planners can generate complex and dynamic whole-body motions [5], [6] and ID controllers nowadays enjoy very fast rates thanks to efficient rigid-body dynamics algorithms [7], [8]. For instance the task function approach [9] which consists in solving IK/ID as the optimization of an operational space task metric can be solved in far less than $1ms$ [10], [11]. However when these planners are not fast enough to allow online replanning, it becomes an issue in face of uncertainty such as external disturbances or modelling errors. Conversely, ID controllers are unable to plan because the instantaneous linearization of the robot dynamics leads to a torque selection based solely on the next desired state. From an optimal control perspective ID can be regarded as the singular case of a zero preview horizon [12]. Yet anticipation at the control level is important for complex tasks, e.g. capturability-constrained balancing tasks in locomotion [13] or non-holonomic orientation control tasks through angular momentum [14], where current actions affect future states.

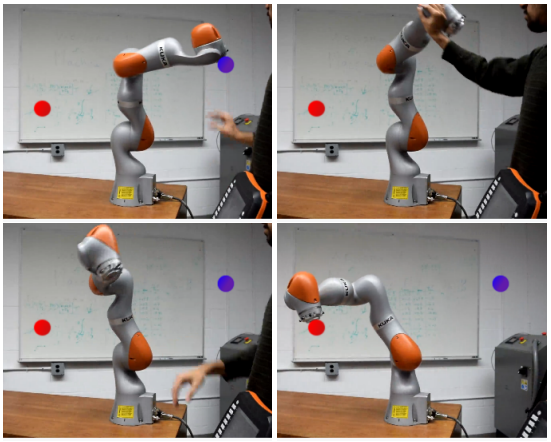


Fig. 1: Pick-and-place task, with high dynamics and precision thanks to our state-feedback MPC, but keeping high compliance thanks to the low-level torque control.

Another approach is to reduce the complexity of the MPC problem by reasoning over simplified models. For example in legged locomotion the linear inverted pendulum (LIPM) can be used in a receding horizon controller to generate stable gaits thanks to its linearity and low dimension [15], [16]. But the whole-body dynamics plays an important role in dynamic motions [17], [18] and is difficult to reflect through simplified descriptions: the LIPM assumes a zero-angular momentum, which prevents it from discovering highly dynamic behaviors such as backflips.

Recent progress in trajectory optimization (TO) offers plausible solutions for preview control. The original continuous-time Optimal Control Problem (OCP) is transcripted into a finite dimensional nonlinear program (NLP) which can then be solved using parametric optimization techniques. In particular Differential Dynamic Programming (DDP) [19] has revealed its potential in whole-body motion generation [20] [14]. DDP is an iterative method based on the *shooting* approach [1], i.e it optimizes over controls while keeping states implicit, as opposed to *collocation* methods [21] which treat states explicitly as decision variables. While

[1]Tandon School of Engineering, New York University, Brooklyn, NY
[2]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[3]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[4]Artificial and Natural Intelligence Toulouse Institute, France

This work was in part supported by the European Union's Horizon 2020 research and innovation program (grant agreement 780684 and European Research Councils grant 637935) and the National Science Foundation (grants 1825993, 1932187, 1925079 and 2026479).

collocation methods can produce impressively realistic behaviors such as in [22] thanks to a proper globalization strategy, they are much slower than shooting methods which exploit the sparsity of the problem. Recent improvements based on multiple-shooting [23] such as FDDP [24] mitigate the poor globalization by allowing unfeasible initialization.

Hardware MPC implementations based on DDP can be found in the literature such as in [25] where the authors propose online replanning based on a DDP-like algorithm where an optimal open-loop torque is combined with a closed-loop PD controller, but sensory feedback is not directly used in the replanning loop. In [26], the authors use iLQR in a separate thread to replan with state feedback every $50ms$ while the robot is controlled at $5ms$. In [27] where the authors propose to accommodate a low re-planning frequency $(15-20Hz)$ with a control frequency of $400Hz$ through frequency shaping. While the proposed controller exhibits good performance it only solves kinematic TO and torques are retrieved through ID. Until the point where the MPC can be evaluated at the frequency of the motor driver inputs, an additional layer of control would be necessary and the MPC would act as a planner.

In this paper we present an MPC controller that computes torques at $1kHz$ over a time horizon based on the whole-body dynamic model. We propose an original optimal-control formulation of the pick-and-place task, and we use FDDP [24] to solve the OCP in an MPC fashion with 2 variants: an open-loop controller similar to [25] computing a feed-forward torque combined with a joint impedance controller, and a closed-loop controller updating the feed-forward torque based on sensory feedback. We show experimentally that the closed-loop controller outperforms the open-loop one. We also compare our controller with ID and kinematic TO. To the best of our knowledge, it is the first experimental demonstration of nonlinear MPC at $1kHz$ with pure torque control on a manipulator without any additional stabilizing controller. The outcomes are of strong practical interest: the robot is able to versatily adapt to perturbation during a dynamic task, reach the target with arbitrary precision *and* offer a full compliance when e.g. an operator physically interacts with it. This paves the road to a new way of designing controllers of manipulator robots in applicative contexts. In Section II the OCP is formulated and solved through DDP-based MPC. In Section II we present our formulation of the cyclic pick-and-place task, which is then benchmarked on the real robot in Section IV.

II. DDP-BASED MPC

In this section we formulate the OCP and solve it in receding horizon fashion using DDP with 2 variants, and we formulate the pick and place task.

A. OCP formulation

The problem is formulated by the continuous-time OCP

$$\min_{u(.),x(.)} \int_0^T L(x(t),u(t),t)dt + L_T(x(T)) \quad (1)$$
$$\text{s.t.} \begin{cases} \dot{x}(t) = F(x(t),u(t)) \quad \forall t \in [0,T] \\ x(0) = x_0 \\ x(t) \in \mathcal{X}, u(t) \in \mathcal{U} \end{cases}$$

where $x = (q,v)$ is the robot's state including position and velocities, $u = \tau$ is the control torque, F the robot's dynamics, L, L_T the running and terminal costs, $\mathcal{X}$,$\mathcal{U}$ are the sets of admissible states and controls.

B. DDP resolution

For more details about the original DDP algorithm we refer the reader to [20] [19]. The time horizon is divided into sub-intervals $[t_i, t_{i+1})$ of length $\delta > 0$ and the solution space is restricted to piecewise constant open-loop controls $U = \{u_0, ..., u_{N-1}\}$. The transcription of (1) reads

$$\min_{X,U} = \sum_{i=0}^{N-1} l_i(x_i,u_i) + l_N(x_N) \quad (2)$$
$$\text{s.t.} \begin{cases} x_{i+1} = f(x_i,u_i) \quad \forall i \in \{0,...,N-1\} \\ x_0 = x_0 \end{cases}$$

where f the Euler or RK4 integration of F and $l_i = \int_{t_i}^{t_{i+1}} L(x_i,u_i)dt$. DDP solves (2) given an initial guess (X^0, U^0) using linear-quadratic (LQ) approximations of f, l and Bellman's recursion by iterating 2 stages

1) *Backward pass* : a quadratic model of the Hamiltonian around the current guess is propagated starting from the terminal node and the corresponding LQRs are solved recursively for $i = N, ..., 0$
2) *Forward pass* : the resulting LQR policy is simulated and the current guess is updated

The inequality constraints are not explicitly taken into account in the solver but implicitly using cost penalization. The algorithm returns optimal state sequence and control policy (X^*, U^*) with feedforward control k^* and feedback gains K^*

$$u_i^*(x) = k_i^* + K_i^*(x - x_i^*) \quad \forall i \in \{0,...,N-1\} \quad (3)$$

The specificity of the FDDP solver we use here is to accept unfeasible initialization to enhance classical DDP's globalization strategy. This is done by using a multiple-shooting transcription of (2), which forces the discretized state trajectory to match the shooting nodes. Concretely, the equality constraint in (2) is changed into $x_{i+1} = f(x_i,u_i) - \bar{f}_{i+1}$ where $\bar{f}_{i+1}$ represent the gap between the integrated state trajectory over $[t_i, t_{i+1})$ and the next shooting node x_{i+1}, i.e. $\bar{f}_{i+1} = f(x_i,u_i) - x_{i+1}$. The forward pass is modified by using a partial nonlinear rollout and the backward pass is modified accordingly. Detailed explanations of the FDDP solver can be found in [24].

C. MPC control schemes

Problem (2) is solved up to an horizon $T_h = N_h\delta$ which returns an optimal state trajectory and control policy (X^*, U^*) as described in Eq. (3). For the new initial guess at the next MPC cycle we use $(X^0, U^0) \leftarrow (X^*, U^*)$ where x_0^* is replaced by the current initial state x_0.

1) Open-loop MPC: In this scheme, planning and control are separated and the plan is updated based on the MPC predictions as shown in Figure 2. The control policy is

$$\begin{aligned} u(\hat{x}) &= u_0^*(x_1^*) + PD(x_1^*, \hat{x}) \\ &= k_0^* + K_0^*(x_1^* - x_0^*) - K_P(\hat{q} - q_1^*) - K_D(\hat{v} - v_1^*) \end{aligned} \tag{4}$$

where $\hat{x} = (\hat{q}, \hat{v})$ is the measured/estimated state. Since planning and control are synchronous, x_0^* is reset to x_1^* at each cycle so the feedback term vanishes and (4) reduces to

$$u(\hat{x}) = k_0^* - K_P(\hat{q} - q_1^*) - K_D(\hat{v} - v_1^*) \tag{5}$$

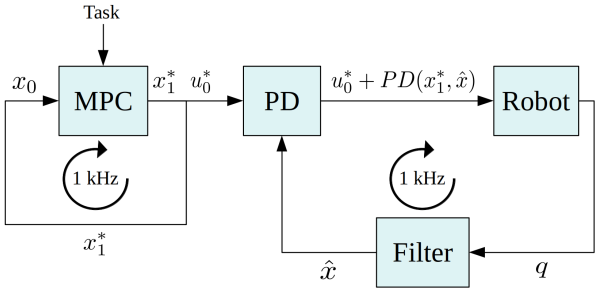


Fig. 2: Open-loop MPC: replan based on predictions. This control scheme is representative of MPC controllers found in the literature such as in [25].

2) Closed-loop MPC: This scheme uses a direct state feedback in the planning loop as shown in Figure 3, hence

$$\begin{aligned} u(\hat{x}) &= u_0^*[x_1^*] \\ &= k_0^* + K_0^*(\hat{x} - x_0^*) \end{aligned} \tag{6}$$

Since the $x_0 = x_0^*$ is reset to $\hat{x}$ at each cycle, the control law reduces to the sole feedforward gain

$$u(\hat{x}) = k_0^* \tag{7}$$

The feedback gain K_0^* compensates small deviations around x_0^* and can be used to interpolate the control trajectory over $[t_0, t_1)$ when the planning rate is lower than the control rate. But as seen from equation (7) the feedback term is not used when planning and control are synchronous.

III. Pick and place task formulation

We propose here an original formulation of the cyclic pick and place task. First we introduce an acyclic reaching task.

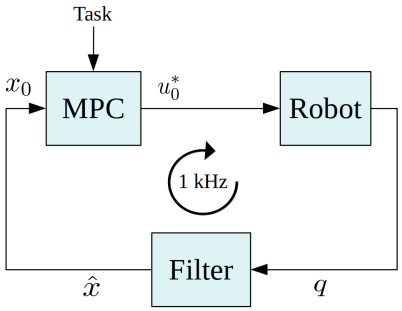


Fig. 3: Closed-loop MPC: direct feedback from sensors. This control scheme is representative of an "ideal" MPC.

A. Static pose reaching task

The task is to bring the end-effector position $p(x)$ to a desired end-effector position $\bar{p}$. The running cost is split into a goal tracking term, regularization terms and a barrier term

$$\begin{aligned} l(x,u) &= l^1(x) + l^2(x) + l^3(u) + l^4(x) \\ &= w\|p(x) - \bar{p}\|^2 + x^T Q x + u^T R u + B(x) \end{aligned} \tag{8}$$

And the terminal cost is similarly defined as

$$\begin{aligned} l_N(x) &= l_N^1(x) + l_N^2(x) + l_N^3(x) \\ &= w_N\|p(x) - \bar{p}\|^2 + x^T Q_N x + B_N(x) \end{aligned} \tag{9}$$

where the weights $w, w_N > 0$ penalize the deviation from the end-effector goal, the weight matrices $Q, R, Q_N, R_N \succ 0$ regularize states and controls and $B(.), B_N(.)$ are weighted quadratic barriers enforcing the state limits $(\underline{x}, \bar{x})$ (i.e. joint position and velocity limits)

$$B(x) = \begin{cases} 0 & \text{if} \quad \underline{x} \le x \le \bar{x} \\ (x - \underline{x})^T B (x - \underline{x}) & \text{if} \quad x \le \underline{x} \\ (\bar{x} - x)^T B (\bar{x} - x) & \text{if} \quad x \ge \bar{x} \end{cases} \tag{10}$$

Note that the OCP for this acyclic task needs to be defined only once as the weights are fixed. Typically a high terminal cost is set on the end-effector goal ($w_N >> w$).

B. Pick and place task

The pick and place task consists in alternatively reaching desired end-effector positions p_a and p_b at predefined times characterized by the cycle duration T_C. The task is enforced by a cost function similar to Eqs. (8),(9), the main difference being that w becomes an increasing sequence of weights over each half-cycle and $\bar{p}$ is modified in order to alternatively penalize targets p_a and p_b. As shown in Figures 2 and 3 the OCP is solved at the control rate ($1kHz$) while the problem formulation (2) only allows an update of the cost function weights at the OCP sampling rate δ. In order to mitigate discontinuities in the solution, the OCP can be updated at each control cycle t. We also add to l^1 a term penalizing the end-effector velocity in order to ensure that the robot comes to a rest each time p_a or p_b is reached

$$l^1(x,t) = w(t)\|p(x) - p^{ab}(t)\|^2 + v(t)\|\dot{p}(x)\|^2 \tag{11}$$

where $w(t), v(t), p^{ab}(t)$ are designed to encode the cyclic nature of the pick and place task, i.e. the alternation of goals p_a and p_b and a zero end-effector velocity at switching times (every $0.5T_C$)

$$w(t) = w^a(t) + w^b(t) \tag{12}$$

$$p^{ab}(t) = \frac{1}{w(t)}(w^a(t)p^a + w^b(t)p^b) \tag{13}$$

$$v(t) = \epsilon w(t) \tag{14}$$

where $w^a(t), w^b(t) > 0$ penalize the distances to p_a, p_b and $\epsilon > 0$ is a scaling factor. Without going into algorithmic details we explain here how these weights are calculated. Their time profile is a periodic function $h(t)$ that is monotonically increasing over each half-cycle

$$h(t) = \int_t^{t+\delta} e^{\alpha(\tau-\beta)}d\tau \quad \forall t \in [k\frac{T_C}{2}, (k+1)\frac{T_C}{2}] \tag{15}$$

where the parameters $\alpha, \beta > 0$ control the slope and sharpness of the exponential. The weights $w^a(t), w^b(t)$ are defined from $h(t)$ such that they are anti-cyclic as shown in Figure 4

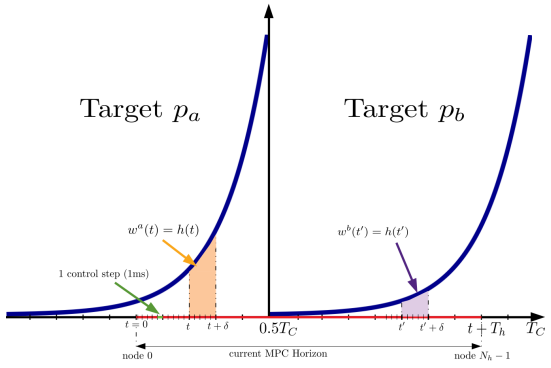


Fig. 4: Weights profile for pick and place task.

IV. Experiments

In this section we implement the controllers introduced in Section II-C on the 7-DoF torque-controlled KUKA LWR iiwa R820 14 shown in Figure 1 for the pick and place task described in Section III. We also compare the closed-loop MPC's performance with ID and kinematic TO.

A. Setup

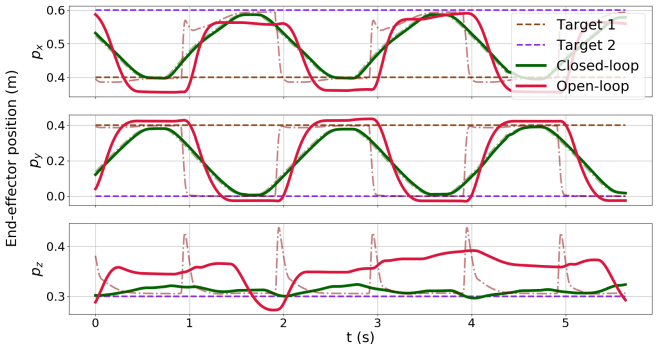
The FDDP solver used in our experiments is available in the Crocoddyl[1] library [24]. Crocoddyl benefits from automatic differentiation and analytical rigid-body dynamics from Pinocchio[2] [8] as well as multi-threading options. Our MPC controllers were implemented in C++ and the real-time data flow was handled using the Dynamic Graph library[3]. The Fast Robot Interface (FRI) extension of the KUKA Sunrise Workbench (v1.3) was used to interface the robot controller with Dynamic Graph. We used an Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz CPU running on RT-Preempt. The KUKA controller applies gravity compensation by default, the actual torque sent to the robot was

$$\tau = u_0^* - g(x_1^*)$$

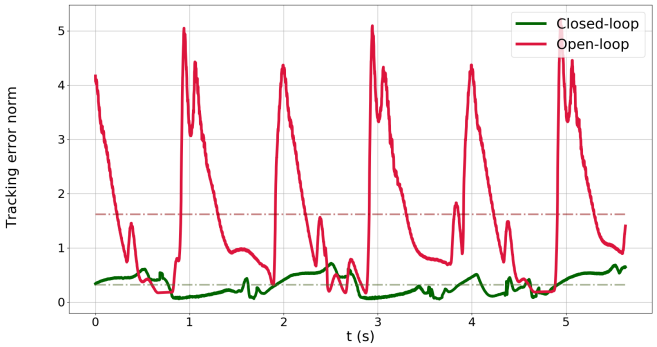
The actuator's rotor inertia was estimated to $0.1kg.m^2$ and added to the diagonal of the inertia matrix.

B. Open-loop vs closed-loop MPC

The regularization weights Q were set to 10^{-2} for the joint positions and 10^{-1} for the joint velocities, the control regularization R to 10^{-2} and the state limit weight B to 50. The running cost weights are defined according to equation (15) with $T_C = 2s$, $\alpha = 40$, $\beta = 0.85$ and $\epsilon = 0.02$. The MPC parameters were set to $\delta = 30ms$ and $N_h = 30$ (i.e. we optimize over a $0.9s$ horizon). Figure 5 shows the end-effector trajectories and tracking errors between MPC predictions and measurements for both controllers. We can see that the open-loop controller generates aggressive desired trajectories requiring unreasonably high PD gains to be tracked. Furthermore it is less optimal for the task as the total running cost was 31934 (against 19978 for the closed-loop controller). In order to improve the tracking performance, the cost function weights can be reduced and the MPC horizon increased for the open-loop controller but this results in



(a) End-effector trajectories (desired in thin dashed line).



(b) 2-norm of the tracking error $\hat{x} - x_1^*$ and average (thin dashed)

Fig. 5: The closed-loop controller generates smooth profiles that are well tracked (in green) while the open-loop controller generates abrupt switches that are impossible to track (in red). High PD gains render the robot stiff and the controller close to instability.

[1]https://github.com/loco-3d/crocoddyl
[2]https://github.com/stack-of-tasks/pinocchio
[3]https://github.com/stack-of-tasks/dynamic-graph

a poor goal tracking precision and the MPC solving time exceeding $1ms$. As a result we had to modify the task and trade-off a significant part of the task precision against compliance in order to improve the tracking, as shown in Figure 6. In conclusion the trade-off between MPC tracking

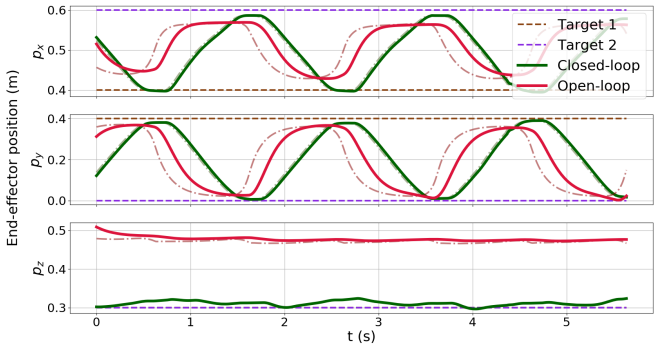
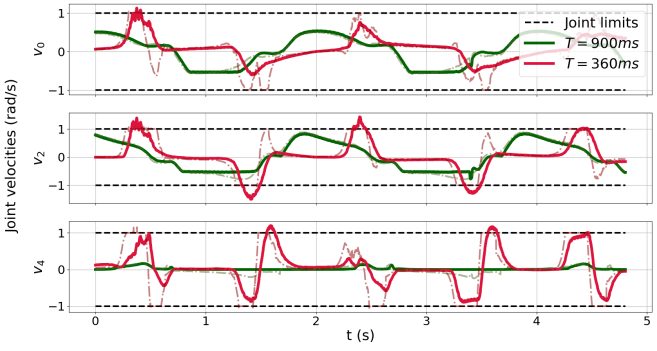


Fig. 6: Reducing the cost weights by setting $\alpha = 20$, $\beta = 0.8$, the open-loop MPC (red) results in a smaller tracking error (average norm 0.7 vs 1.6 in Figure 5) but a loss in the end-effector precision, as seen by comparison with the original closed-loop MPC trajectory (green) duplicated from Figure 5.

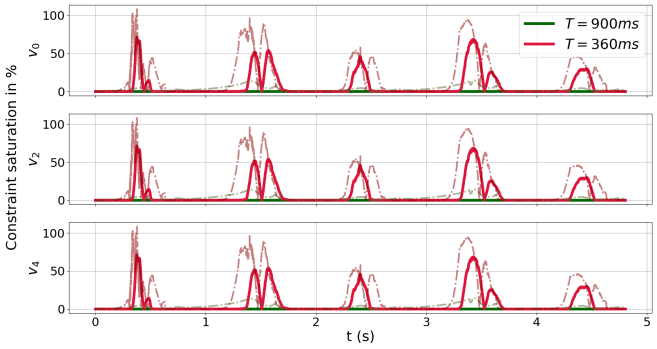
and task fulfillment appears challenging to arbitrate with the open-loop controller since it leads either to a high gains control loop close to instability, or to a poor task performance with a longer solving time. This experimental data shows that the open-loop control scheme is not suited for online use as the predictions do not match the actual capabilities of the robot. This confirms that the state feedback is crucial to generate online smooth and realistic trajectories. The attached video shows the disturbance rejection performance of each controller. The stiff open-loop MPC compensates poorly an external push whereas the closed-loop controller exhibits a more compliant, yet effective, response.

C. Comparison with ID

When the horizon of an OCP collapses, the trajectories consists in a single point optimized under the instantaneous dynamics constraints and it is equivalent to ID, as shown in Appendix . As a consequence we expect to observe a reduced anticipation ability of the closed-loop controller as $T_h \to 0$. In order to exhibit this phenomena and emphasize the importance of the horizon length to anticipate constraints, we look at the joint velocity constraint saturation in the pick and place task. For the same task as in the first experiment, Figure 7 shows joint velocities obtained for $T_h = 900ms$ and $T_h = 360ms$ as well as the velocity constraints saturation and Figure 8 shows the average saturation over all joints as a function of T_h. As expected when $T_h \to 0$, the velocity constraint saturation increases which confirms that short horizons are less able to anticipate constraints. This resulted in a higher running cost (14182 against 10048) so the shorter horizon is less optimal for the task. This experimental data shows that a sufficiently long horizon in control schemes is important to avoid myopic behaviors.



(a) Joint velocities (desired in thin dashed line)



(b) Joint velocities saturation (desired in thin dashed line)

Fig. 7: The velocity constraint saturation peaks more significantly at each half-cycle when the horizon is smaller: short-sighted controllers are less able to anticipate constraints.

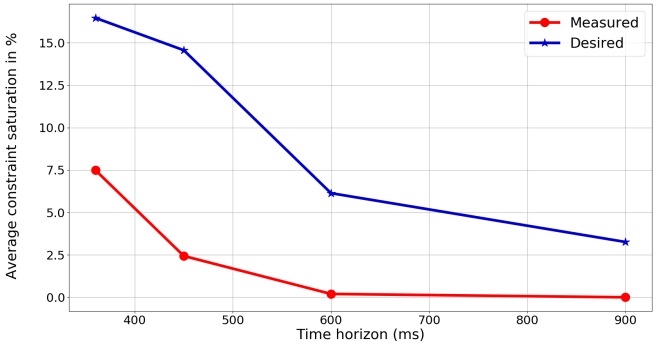
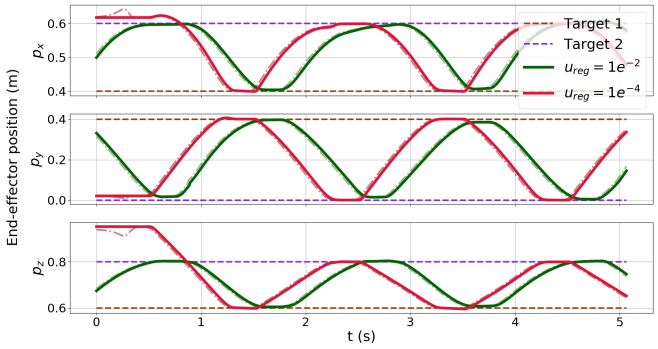


Fig. 8: Average constraint saturation over all joints as a function of the time horizon. Shorter horizon controllers hit the constraint more often.

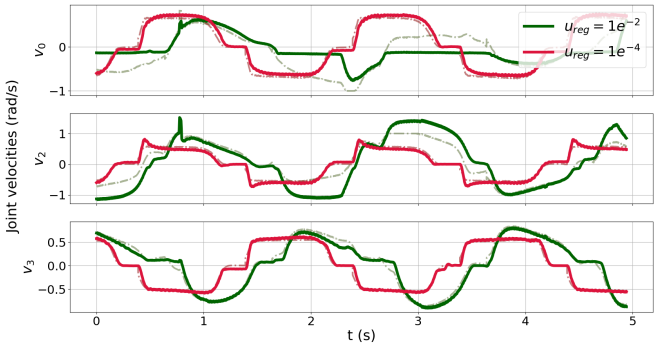
D. Comparison with kinematic TO

Taking into account the full dynamics in the preview horizon is important in order to generate dynamically consistent motions. This can be verified by reducing the control regularization: when $R \to 0$, the optimizer should neglect the dynamics and return kinematically optimized trajectories that are dynamically not relevant. Indeed, for a fixed-based manipulator, any desired accelerations can be realized through ID at the cost of potentially very high torques. Figure 9 shows the end-effector positions, joint velocities and torque profiles obtained for $R = 10^{-2}$ and $R = 10^{-4}$. As expected when the

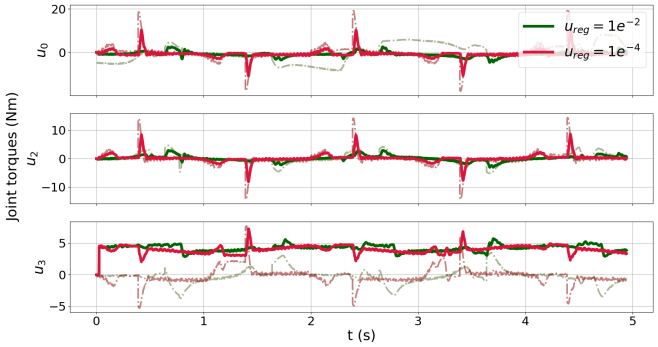
control is less penalized, the MPC generates abrupt torque and velocity switches that are not consistent with the robot's mechanical capabilities.



(a) End-effector trajectories and reference (thin dashed line).



(b) Estimated joint velocities and reference (thin dashed line)



(c) Filtered joint torques and reference (thin dashed line)

Fig. 9: When the control regularization term decreases, the trajectories are less relevant: the velocity tends toward a square signal and the torque exhibits sharp peaks

V. Conclusion

We introduced the first whole-body torque MPC controller running at $1kHz$ on a real robot. We showed experimentally that this controller is able to achieve with good performance dynamic cyclic tasks while ensuring a compliant response to external disturbances. It outperforms state-of-the art MPC schemes that lead to stiff behaviors and less accurate goal tracking. We also exhibited the importance of the preview horizon and full dynamics in the performance and thereby its superiority against kinematic optimization and ID which is merely a singular case of MPC with a collapsed horizon. As future work we would like to improve our controller's efficiency in order to increase the preview horizon and achieve more dynamic tasks, and to extend this approach to more complex scenarios involving contacts with the environment, with the ultimate objective of applying it to multi-contact locomotion.

Appendix

We show here that ID is equivalent to an OCP of which the horizon collapses. Task-Space Inverse Dynamics (TSID) consists in computing the joint torques achieving a desired task-space impedance and can be written as a QP

$$\begin{aligned} \min_{\tau} \ & \alpha_1 \|J(q)a + PD_1(p(q), \dot{p}(v)\|^2 \\ & + \alpha_2 \|a + PD_2(q, v)\|^2 + \alpha_3 \|\tau\|^2 \\ \text{s.t. } & a = M^{-1}(\tau - b) \end{aligned} \tag{16}$$

where q, v, a are the joint positions, velocities and accelerations, $p(q)$ is the end-effector position, $J = \frac{\partial p}{\partial q}(q)$ is the Jacobian, PD_1, PD_2 are task-space and joint-space impedances, $b = b(q, v)$ summarizes Coriolis, gravity and centrifugal effects. Consider the OCP (2) when its horizon collapses ($N = 1$)

$$\begin{aligned} \min_{u_0, x_0, x_1} \ & l_0(x_0, u_0) + l_1(x_1) \\ \text{s.t. } & \begin{cases} x_1 = f(x_0, u_0) \\ x_0 = x_0 \end{cases} \end{aligned} \tag{17}$$

Selecting a cost function encoding a task-space objective with joint-space regularization, e.g.

$$\begin{aligned} l_0(x, u) &= \|p(q)\|^2 + \|Jv\|^2 + \|q - \bar{q}\|^2 + \|v\|^2 + \|u\|^2 \\ l_1(x) &= \|p(q)\|^2 + \|Jv\|^2 + \|q - \bar{q}\|^2 + \|v\|^2 \end{aligned} \tag{18}$$

where $\bar{q}$ is a reference posture. Since DDP uses the shooting approach, the state is not an explicit decision variable so the singular-horizon OCP (17) is equivalent to

$$\min_{u_0} \ \|u_0\|^2 + \|p(q_1)\|^2 + \|Jv_1\|^2 + \|q_1 - \bar{q}\|^2 + \|v_1\|^2 \tag{19}$$

$$\text{s.t. } a_1 = M^{-1}(q_0)(\tau - b(q_0, v_0))$$

The Euler 2^{nd}-order integration with step $\delta << 1$ leads to

$$\begin{aligned} p(q_1) &\simeq p(q_0) + Jv_0\delta + Ja_1\delta^2 \\ Jv_1 &\simeq Jv_0\delta + Ja_1\delta \end{aligned} \tag{20}$$

and using (20) in (19), it can be shown that the collapsed OCP can be written under the form

$$\begin{aligned} \min_{\tau} \ & \beta_1 \|Ja + K_1 p(q) + K_2 Jv\|^2 \\ & + \beta_2 \|a + K_3(q - \bar{q}) + K_4 v\|^2 \end{aligned} \tag{21}$$

$$\text{s.t. } a = M^{-1}(\tau - b) \tag{22}$$

where $(K_1, K_2), (K_3, K_4), \beta_1, \beta_2$ are task-space PD gains, joint-space PD gains, and weights all depending on δ (and the OCP cost weights if any). This problem is equivalent to (16), which concludes the proof that TSID is merely the particular case of an OCP with a singular horizon.

REFERENCES

[1] M. Diehl, H. G. Bock, H. Diedam, P.-b. Wieber, M. Diehl, H. G. Bock, H. Diedam, P.-b. W. Fast, and D. Multiple, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," p. 28, 2009.

[2] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.automatica.2014.10.128

[3] V. Duchaine, S. Bouchard, and C. Gosselin, "Computationally efficient predictive robot control," *Mechatronics, IEEE/ASME Transactions on*, vol. 12, pp. 570 – 578, 11 2007.

[4] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[5] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On Time Optimization of Centroidal Momentum Dynamics," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5776–5782, 2018.

[6] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[7] R. Featherstone, *Rigid Body Dynamics Algorithms*, 2008.

[8] 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," *Proceedings of the 2019 IEEE/SICE International Symposium on System Integration, SII 2019*, pp. 614–619, 2019.

[9] L. Saab, O. E. Ramos, F. Keith, N. Mansard, P. Soueres, and J. Y. Fourquet, "Dynamic whole-body motion generation under rigid contacts and other unilateral constraints," *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 346–362, 2013.

[10] A. Del Prete, N. Mansard, O. Ramos, O. Stasse, and F. Nori, "Implementing torque control with high-ratio gear boxes and without joint-torque sensors," *International Journal of Humanoid Robotics*, vol. 13, 03 2015.

[11] A. Herzog, N. Rotella, S. Mason, F. Grimminger, S. Schaal, and L. Righetti, "Momentum control with hierarchical inverse dynamics on a torque-controlled humanoid," *Autonomous Robots*, vol. 40, no. 3, pp. 473–491, 2016.

[12] P. Geoffroy, N. Mansard, M. Raison, S. Achiche, and E. Todorov, "From Inverse Kinematics to Optimal Control," *Advances in Robot Kinematics*, pp. 409–418, 2014.

[13] P. B. Wieber, "Viability and predictive control for safe locomotion," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 1103–1108, 2008.

[14] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics," 2018.

[15] 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," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1620–1626, 2003.

[16] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic foot step placement," *Advanced Robotics*, vol. 24, pp. 719–737, 04 2010.

[17] D. E. Orin, A. Goswami, and S. H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[18] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast motions in biomechanics and robotics*. Springer, 2006, pp. 411–425.

[19] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[20] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," *IEEE International Conference on Intelligent Robots and Systems*, pp. 4906–4913, 2012.

[21] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[22] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, 2012.

[23] H. Bock, M. Diehl, J. Schlöder, F. Allgöwer, R. Findeisen, and Z. Nagy, "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *IFAC Proceedings Volumes*, vol. 33, pp. 671–679, 06 2000.

[24] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," no. March, pp. 2536–2542, 2020.

[25] M. Neunert, F. Farshidian, A. W. Winkler, and J. Buchli, "Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1502–1509, 2017.

[26] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body Model-Predictive Control applied to the HRP-2 Humanoid Bennewitz , Nicolas Mansard," 2015.

[27] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for Torque-Controlled Legged Robots," *IEEE International Conference on Intelligent Robots and Systems*, no. May, pp. 4730–4737, 2019.