Whole Body Model Predictive Control with a Memory of Motion: Experiments on a Torque-Controlled Talos

Ewen Dantec [a,b,*], Rohan Budhiraja [a], Adria Roig [c], Teguh Lembono [d], Guilhem Saurel [a], Olivier Stasse [a,b], Pierre Fernbach [e], Steve Tonneau [f], Sethu Vijayakumar [f], Sylvain Calinon [d], Michel Taix [a], Nicolas Mansard [a,b]

***Abstract*— This paper presents the first successful experiment implementing whole-body model predictive control with state feedback on a torque-control humanoid robot. We demonstrate that our control scheme is able to do whole-body target tracking, control the balance in front of strong external perturbations and avoid collision with an external object. The key elements for this success are threefold. First, optimal control over a receding horizon is implemented with Crocoddyl, an optimal control library based on differential dynamics programming, providing state-feedback control in less than 10 ms. Second, a warm start strategy based on memory of motion has been implemented to overcome the sensitivity of the optimal control solver to initial conditions. Finally, the optimal trajectories are executed by a low-level torque controller, feedbacking on direct torque measurement at high frequency. This paper provides the details of the method, along with analytical benchmarks with the real humanoid robot Talos.**

A video of the experiment is available at https://peertube.laas.fr/videos/watch/cbc25927-337c-4635-a1bc-153b9aeb4135

I. Introduction

Model predictive control (MPC) [1] has become a widely used tool in the robotics community because of its ability to handle non-linearity, constraints and system dynamics [2]. MPC formulation has been implemented for a great number of robotics and industrial applications: torque control of 4-legged robot [3], path planning for autonomous vehicle [4], quadrotor control [5] or industrial processes [6].

MPC typically involves the prediction of the future robot behaviour over a receding finite-time horizon, based on a mathematical model of the system dynamics [7], [8]. At each control cycle, the state of the system is estimated and a new optimal control problem (OCP) is solved, starting from the state estimate, for a finite horizon in the future shifted from one sampling period. This OCP typically features a non-linear cost function describing the tasks to achieve and the constraints to satisfy. Application of MPC in robotics has yet been limited by computational limits, as complexity scales at least with the cube of the system dimension. Moreover, non-linearities in the robot model make the OCP non-convex, which first increases the computational load but may also trap the MPC in improper local minimum [9]. MPC has thus been mostly applied to systems with small numbers of degrees of freedom [10], [11], [12] or to reduced dynamics [13], [14]. First attempts in the direction of controlling the robot from an OCP have led to hierarchical control architectures where an OCP solver is used at low frequency to replan the trajectory which is then tracked by a dedicated controller [10].

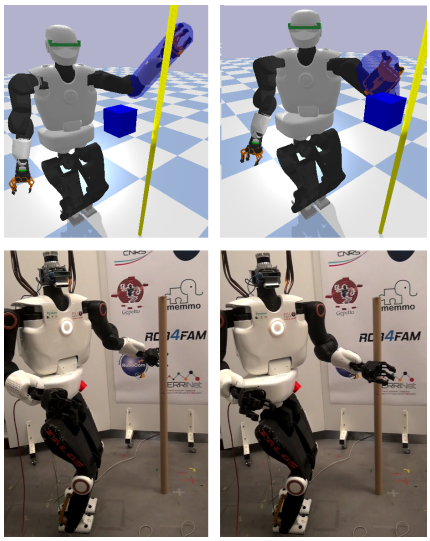

Fig. 1: The humanoid robot Talos 1 performing reactive collision avoidance while following a moving target, driven by a MPC at 100 Hz.

Several challenges have to be tackled in order to use a whole-body MPC directly as the main robot controller. Independently from the particular OCP formulation, we first need an efficient numerical optimal control algorithm able to work at high frequency [15], [16], [17]. Direct transcription methods [18] [19], in which both state and control are treated as variables, are common to solve such problem. Others transcriptions of the OCP are now considered to be more suitable to MPC, in particular shooting methods like Differential Dynamics Programming (DDP) [20], [21]. Efficient solvers based on DDP have already been implemented [22], [23] and are able to deal with non linear costs, constraints and dynamics.

A second challenge is to reduce the number of iterations needed by the solver to converge. This can be done either by formulating the DDP as a more convex problem [24], or by providing to the solver a good initial guess. A conventional

[a] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[b] Artificial and Natural Intelligence Toulouse Institute, France
[c] PAL Robotics, Barcelona, Spain
[d] Idiap Research Institute, Switzerland
[e] TOWARD, Toulouse, France
[f] IPAB, The University of Edinburgh, Scotland
[*] corresponding author: edantec@laas.fr

This work is supported by the European project MEMMO (GA-780684) and the FLAG-ERA JTC 2016 RobCom++ project.

heuristic in non-linear MPC consists in building such a warm start from the solution computed at the previous control cycle [25]. This is a good heuristic but may fail in front of sudden disturbances. A more generic solution has been proposed by learning offline some approximation of the optimal behavior. In [26] such an initialization is learned by reinforcement learning on simple systems. In [27], several regression algorithms for learning a good warm start from some pre-computed database of motions are compared in the context of locomotion. We refer here to such a learned initialization as *memory of motion*.

Finally, a third challenge is to decide at which level the MPC should be connected to the lower servo controller of the robot. In our opinion, it should be connected as low as possible, yet this implies that the MPC has to be evaluated at a compatible high frequency. In particular, if the MPC should directly work with a low-level torque controller [28], [29], it implies it has to be updated every 10 ms (to maintain a torque control frequency of 100 Hz at least).

In this paper, we propose a transversal answer to these three challenges and report the first complete demonstration of a whole-body MPC on the torque-controlled humanoid robot TALOS [30]. We performed a reactive manipulation task involving 28 degrees of freedom, contacts, balance and collision avoidance over a frequency rate of 100 Hz. To this end, we relied on several key components: first, an efficient OCP solver, tailored for robot dynamics, enables us to evaluate the MPC at 100 Hz. Then, a memory of motion is used in parallel to predict a good warm start and helps the solver to handle non-convexities. The robot servo control implements a torque control running at 2.3 kHz and is able to robustly apply the control references produced by the MPC.

The paper is structured as follows. Section II discusses the basic formulation of optimal control for whole body motion with contact. Section III explains the construction of the memory of motion and how it improves the convergence of DDP. The implementation details are given in Section IV. Section V presents the experimental results on the humanoid robot TALOS.

II. Whole body optimal control with contact

A. Contact-Constrained Rigid body dynamics

Given a rigid body system with n_j joints and P rigid contacts with the environment, the dynamics of this constrained multi-body system is described in [15] and [31] by:

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

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 position 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 motion freedom matrix selecting the actuated joints, $\boldsymbol{\tau}$ the joint torques, $\boldsymbol{\lambda} = [\boldsymbol{\lambda}_1 \cdots \boldsymbol{\lambda}_P]^T$ and $\boldsymbol{J}_c = [\boldsymbol{J}_1 \cdots \boldsymbol{J}_P]^T$ 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:

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

We then classically consider the state of the robot to be $\boldsymbol{x} = (\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the control to be $\boldsymbol{u} = \boldsymbol{\tau}$. In [31], we showed that with this formulation, forces become consequences of the state $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and the control $\boldsymbol{\tau}$, and thus our dynamics becomes independent from these variables. Consequently, (1) leads to the following force-free partial derivative equation:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{q}} \end{bmatrix} = f(\boldsymbol{x}, \boldsymbol{u}), \tag{3}$$

where $\ddot{q}$ is deduced from $(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{\tau}$ using (1).

B. Optimal control for legged robots

Based on (3), we can formulate the OCP as:

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

where T is the horizon time, over which we want to make our prediction; 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 optimal control problem for locomotion often contains admissibility constraints on state and control. However, OCP with constraints are difficult (and slower) to solve. Thus, we formulate these constraints as penalties of various forms in our cost function, and describe them in Section IV.

OCP (4) should be transcripted into a nonlinear program in order to be solved. We used a multiple-shooting formulation, as it combines efficiency of shooting formulations [32] with numerical stability [33].

C. Solving the OCP with DDP

DDP provides a fast and efficient way to numerically solve a discretized transcription of (4). As we used a multiple shooting transcription, we quickly describe here the classical DDP and how it is modified to handle it.

At each solver iteration the OCP is discretized with time step $\delta t = \frac{T}{N}$ and then linearized around an initial discretized trajectory $(\boldsymbol{x_t}, \boldsymbol{u_t}), \forall t = \{0, \cdots, N-1\}$. This problem then

becomes:

$$\min_{\underline{\boldsymbol{\Delta x}},\underline{\boldsymbol{\Delta u}}} \Big(\sum_{t=0}^{N-1} \frac{1}{2} \begin{bmatrix} \boldsymbol{\Delta x}^\top, \boldsymbol{\Delta u}^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{L_{xx}} & \boldsymbol{L_{xu}} \\ \boldsymbol{L_{ux}} & \boldsymbol{L_{uu}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta u} \end{bmatrix} + \begin{bmatrix} \boldsymbol{L_x} & \boldsymbol{L_u} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta x} \\ \boldsymbol{\Delta u} \end{bmatrix} + \frac{1}{2} \boldsymbol{\Delta x}_T \boldsymbol{L_{xx}} \boldsymbol{\Delta x}_T + \boldsymbol{L_x} \boldsymbol{\Delta x}_T \Big) \quad (5)$$

$$s.t. \quad \boldsymbol{\Delta x}_0 = \boldsymbol{f}_0$$

$$\forall t = \{0, \cdots, N-1\},$$

$$\boldsymbol{\Delta x}_{t+1} = \boldsymbol{F_x} \boldsymbol{\Delta x}_t + \boldsymbol{F_u} \boldsymbol{\Delta u}_t + \boldsymbol{f}_{t+1}$$

where $\boldsymbol{L_x}, \boldsymbol{L_u}$ and $\boldsymbol{L_{xx}}, \boldsymbol{L_{xu}}, \boldsymbol{L_{uu}}$ are the gradients and Hessians of the cost function (indices t have been dropped for simplicity). 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.

DDP solves (5) by using Bellman's principle [34], which naturally exploits the sparsity of the Markovian nature of the dynamics constraints. Eventually, each optimal step in a DDP iteration can be produced by an iterative scheme that computes the descent direction in a backward pass (going from N to 0), and then finds the optimal step along this direction in a forward pass (going from 0 to N). While this approach requires that the initial point of linearization is feasible, a multiple-shooting variant by our team, namely FDDP [35], allows us to relax this requirement and start from an infeasible trajectory. For more details, readers are referred to [36] [37] for DDP and [35] for FDDP schemes.

III. Creating the Memory of Motion

A. Building the memory dataset

As the transcription of (4) is non-convex for complex robots, the behavior of the OCP solver cannot be guaranteed: it may get stuck in a poor local minima if the initialization is bad. The MPC is then augmented with an external process to infer candidate initialization. The problem of inferring the warm start is formulated as a regression problem $\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{y}$ where the input task consists of the current robot state and goal location, and the output $\boldsymbol{y}$ consists of the corresponding state and control trajectories. Such a *memory of motion* [27], [38] has been shown to improve the convergence of DDP significantly. It relies on a dataset of optimal trajectories built off-line and encoded by machine learning.

We build the dataset in two steps: the first step uses a sampling-based planner to create an initial dataset of trajectories, ensuring that a solution is obtained if it exists; in the second step we optimize the first dataset using a local optimizer. We used Constrained Bi-directional Rapidly-Exploring Random Tree (CBIRRT) [39], [40] for the first step and the same OCP problems than at run-time, solved by DDP for second step. The dataset is stored in the format $\{\boldsymbol{\theta}_i, \boldsymbol{y}_i\}, \forall i \in \{1, \ldots, N_m\}$ where $N_m = 1000$.

Given a new task $\hat{\boldsymbol{\theta}}$, we use nearest neighbor (NN) algorithm to compute the closest $\boldsymbol{\theta}_i$ to $\hat{\boldsymbol{\theta}}$ in the dataset and output the corresponding $\boldsymbol{y}_i$ as the warm start. We use the Euclidean metric and the efficient K-D tree as data structure [41] for the NN algorithm.

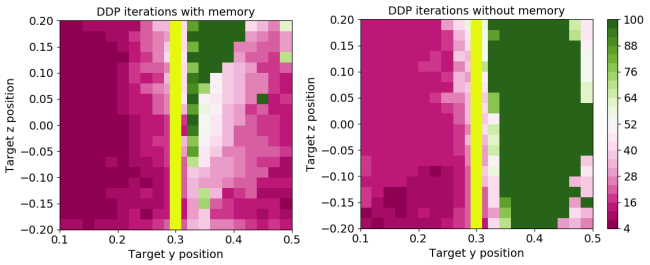

Fig. 2: Comparison of convergence speed of the OCP solver (left) with and (right) without warm-start, for a grid sampling of the target position, while the initial configuration is always the same. The metric is the number of iterations to convergence, with an upper limit of 100 iterations. The arm initial position is on the bottom left corner of each picture. The obstacle (a pole) is highlighted in yellow.

Gaussian Process Regression (GPR) was shown in [27] to perform much better than NN in producing DDP warm starts for unimodal tasks. Yet the task considered in this work is multimodal, i.e., there are several qualitatively different trajectories to achieve one task. For such problem, GPR performs poorly as observed in [38], hence we choose NN in this work instead.

B. Evaluating the impact of memory warm-start

In order to understand how the initialization improves the behavior of the DDP solver in the case of a non-convex problem, let us consider an end-effector reaching task problem involving an obstacle (see 1). In simulation, this OCP has been solved several hundred times with slightly different target positions: while the x position of the targets remains at coordinate 0.07, the y and z positions are bound to go respectively from 0.2 to 0.5 and from -0.2 to 0.2 (in the robot frame). The obstacle is a 2 cm diameter vertical pole set at coordinate $(x = 0.6, y = 0.3)$.

This set of problems is solved in two ways: first with a memory warm-start corresponding to the current target position, secondly with a static initialization identical for all target positions. The number of iterations until convergence has been chosen as a metric to compare the behavior of the solver with and without memory. The results of this experiment is presented in Fig. 2.

As the arm starts from the bottom left corner, the left target configurations are easier to reach. When the target is located at the right side of the obstacle, the solver barely finds a path around it in less than 100 iterations, meaning that it cannot find the optimal solution of the non-convex problem in reasonable time. So when the target is on the wrong side, the MPC absolutely needs the memory initialization. Even when the target is on the good side, the memory is beneficial.

At run time, a good initialization is frequently inferred from the memory, then refined (as examplified by Fig. 2))

and finally provided to the MPC as detailed in the following section.

IV. MPC Implementation with Memory of Motion

A. An OCP for reaching, balancing and avoiding collisions

The OCP is implemented with a running cost composed of the sum of five terms: squared distance to the target, penalization of the CoM drift, penalization enforcing joints limits, state and control regularization. A last term is added to take into account the environment collisions: given a pair of collision between bodies A and B, we write $d_{AB,t}$ as the minimal distance between these two objects at time t. Let us define d_{min} as distance threshold for the activation of the cost, and c_{col}, a weight for the cost. Then the collision penalization in the cost function is written as:

$$l_{\text{col},t} = \begin{cases} c_{\text{col}}(d_{AB}(t) - d_{\text{min}})^2 & \text{if} \quad d_{AB}(t) < d_{\text{min}} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where the distance $d_{AB}(t)$ is computed from the two body placements using standard proximity algorithm [40]. In order to simplify the problem, we only consider collision between the end effector (left arm of the robot) and a fixed obstacle in the environment (see Fig. 1), while modeling both the arm and the obstacle by dedicated capsules.

One could notice that we do not implement friction cone constraints as described in equation (2). This is possible by adding a barrier cost with high weight, but it has not been done in this paper because the performed tasks don't require it, and the robot is able to keep its balance without it.

B. Warm start using temporal coherence and memory

Every 10 ms the OCP is updated by setting the initial state to the latest estimated state $\boldsymbol{x}_m$, and by updating some environment parameters (e.g. position of the target to reach). In order to produce a desired control in less than 10 ms, only one DDP iteration is performed each time the OCP is recomputed, so a warm-start close to the optimal solution is needed. A first way to warm-start the solver is to heuristically update the previous optimal trajectory $\{(\boldsymbol{x}_0^*, \boldsymbol{x}_1^*, \cdots, \boldsymbol{x}_N^*), (\boldsymbol{u}_0^*, \boldsymbol{u}_1^*, \cdots, \boldsymbol{u}_{N-1}^*)\}$. This trajectory is shifted by one knot and $\boldsymbol{x}_m$ is used as the initial state. The resulting warm start follows [25]:

$$\begin{aligned} \{\boldsymbol{x}_{\text{guess}}, \boldsymbol{u}_{\text{guess}}\} = \{&(\boldsymbol{x}_m, \boldsymbol{x}_2^*, \cdots, \boldsymbol{x}_N^*, \boldsymbol{x}_N^*), \\ &(\boldsymbol{u}_1^*, \boldsymbol{u}_2^*, \cdots, \boldsymbol{u}_{N-1}^*, \boldsymbol{u}_{N-1}^*)\}. \end{aligned} \tag{7}$$

This initialization is acceptable for simple problems like reaching a target in a collision-free environment. Yet it can only track the previous optimum if it is continuously moving across time, not if it suddenly jumps e.g. because the target has shifted too far away. We see clearly from Fig. 2 that it becomes problematic when dealing with obstacles, since the robot may get stuck in poor local minima despite the fact that a better trajectory exists.

The memory of motion described in Sec. III is implemented to give to the MPC a look-ahead and provide alternative trajectories when the previous warm-start strategy is not sufficient. Given a target to reach, obstacle positions and state estimate, the memory infers the corresponding state and control trajectory and further refines this candidate by several DDP iterations, using the same DDP solver as for the MPC. In order to output a good refined warm start in a reasonable amount of time (less than 0.1 s), we chose to limit the refinement computation to 10 iterations. Once refined, the cost of the warm start is compared against the cost of the current MPC trajectory. The memory warm start is accepted by the MPC if its cost is lower.

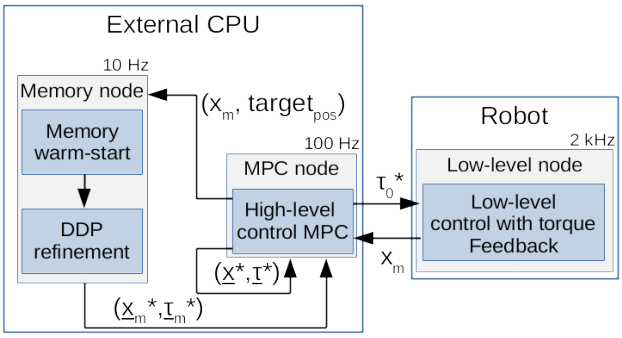

Fig. 3: Diagram of the ROS implementation with all 3 nodes. $(\boldsymbol{x}_m, \text{target}_{\text{pos}})$ are the initial state and current target position sent to the memory node; $(\underline{x}^*, \underline{\tau}^*)$ are the current optimal state and control trajectories produced by the MPC; $(\underline{x}_m^*, \underline{\tau}_m^*)$ are the warm start trajectories computed by the memory of motion..

C. Low-level control

The optimal torques computed by the DDP are decoupled for each joint and sent as reference to the low-level torque control running at 2 kHz. This control is a PD+ that reads the values from the torque sensors and uses an observer to estimate the intrinsic dynamics of the joint which are not considered in the model. The objective is to compensate this unexpected dynamics such as the inertia of the motor, the high friction produced by the high reductions in the mechanical transmission or the inner flexibility of the harmonic-drive, so that it behaves as an ideal joint.

D. ROS architecture

The implementation needs three parallel processes to run simultaneously (see Fig. 3): the MPC, the memory and the low-level torque control. We used one full CPU for each process. ROS publisher-subscriber architecture is used to build the communication framework between all running processes to synchronize their different frequencies. Thus, every ROS node uses the latest available data on the topics it subscribes to, and publishes the output once computation is done.

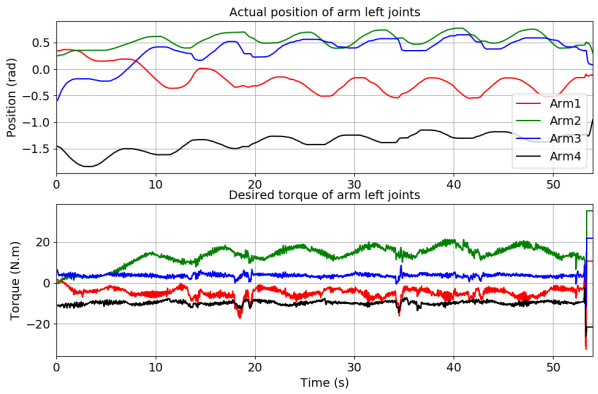

Fig. 4: Positions and torques for the 4 joints of the left arm in whole body experiment.

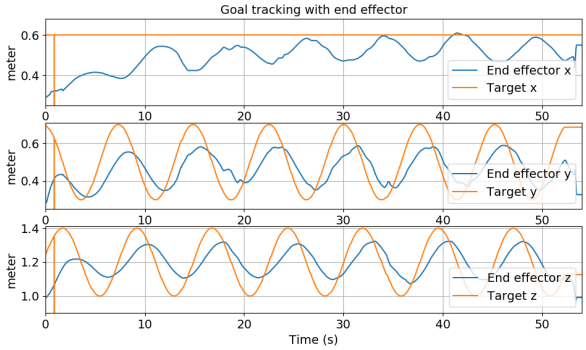

Fig. 5: Tracking of a moving target with the left arm end effector in whole body experiment.

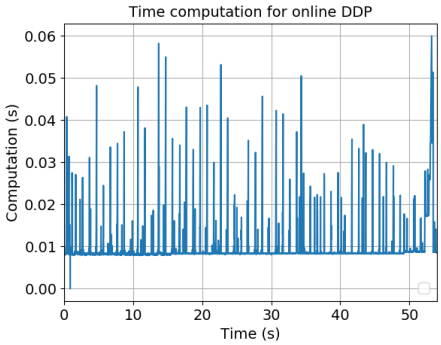

Fig. 6: Time computation of one DDP iteration for the whole body experiment. Mean value is 9.1 ms. The visible spikes are isolated surges in time computation that do not impact the overall behaviour of the system.

V. Experimental results

A. Experimental setup

The experimental setup aims at answering two main questions: can the OCP be solved in real time for non-convex problems, typically path planning with obstacles problems? And how does the memory help to solve non-convex problems? In order to answer these questions, three different experimental protocols were tested on our robot Talos, each of them using the same version of the MPC algorithm.

B. Lower body squat in contact

First, we performed an experiment with only lower body in contact: 12 joints are controlled in torque (legs only) and the DDP cost function takes into account a state and control regularization cost (weights 10^{-2} and 6×10^{-3}), a center of mass regularization cost (weights 10^3) and a joint limit cost (weights 10^3). The CoM target to track goes up and down along the vertical axis, making the robot squat. We could solve this problem in less than 10 ms using 100 shooting nodes. The squat demonstration is shown in the companion video.

C. Whole-body in contact

Secondly, we performed a goal tracking with whole-body in contact experiment. In this setup, 22 joints are controlled in torque (arms, legs and torso) and the DDP cost function takes into account a state and control regularization cost (weights 2×10^{-2} and 4×10^{-4}), a center of mass regularization cost (weights 10^2), a joint limit cost (weights 10^3) and a goal tracking cost (weights 15). The target to track goes along a circle in front of the robot, near enough to be reached. Since this problem is harder to solve, we use only 50 shooting nodes in order to compute the solution in less than 10 ms.

The tracking of the moving target was achieved with the whole body model at 100 Hz. As shown in Fig. 5, the end effector properly follows the target while the robot keeps its balance. Low target tracking gains have been chosen on purpose, in order to reach a compromise between a good tracking and a good balance: indeed, increasing the tracking weight tends to produce stiffer and more brutal control, which can push the robot out of balance.

In Fig. 4, the actual positions and desired torques sent to the low level control by the MPC are displayed. Although desired torques appear noisy (reflecting the noise in state estimation), joint positions are rather smooth, which leads to a clean movement on the robot.

In Fig. 6, the computation time needed to solve the online DDP is displayed. Only some time spikes are at more than 10 ms (because we have chosen to not embed the MPC in a real-time operating system) and the mean value remains below 10 ms. This mean value does not changed significantly when external disturbances are applied, since only one iteration of DDP is performed in any case. Even if some outputs are late from time to time, this does not impact the whole behavior of the robot as the low-level control will simply maintain the last received desired torque while waiting for a new entry.

While running the experiment, the robot was sometimes faced with external perturbations that it handled well (see video). The arms, the legs and the torso were pushed firmly with a stick while the MPC robustly handled the disturbances.

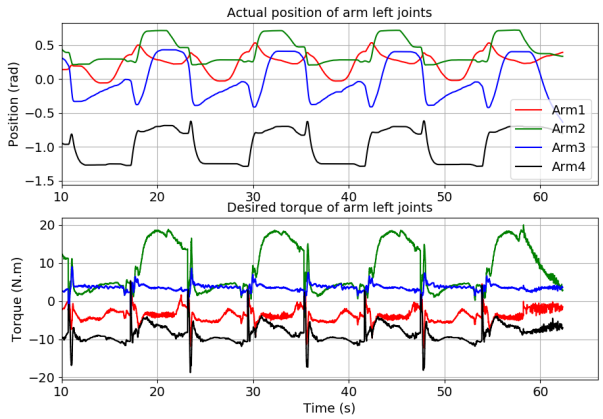

Fig. 7: Positions and torques for the obstacle experiment.

D. Collision avoidance with memory

Finally, we performed a goal tracking and collision avoidance experiment. In this setup, 6 joints are controlled in torque (left arm and torso) and the DDP cost function takes into account a state and control regularization cost (weights 10^{-2} and 10^{-2}), a joint limit cost (weights 10^3), a goal tracking cost (weights 20) and a collision avoidance cost (weights 10^4) involving one pair of collision between the arm and a fixed obstacle in the environment. This time, the target to track goes along a line parallel to the ground and behind a 2 cm wide vertical pole which plays the role of an obstacle (see Fig. 1).

In this setting, the collision-free tracking control was achieved on the robot at a frequency of 100 Hz with 100 shooting nodes. Because this problem is harder to solve, we let the DDP perform at most 5 iterations instead of just one, in order to get better solutions. As can be seen in Fig. 9, the mean time computation of the OCP is way below 10 ms, implying that a working frequency of 200 Hz may be possible to reach for this particular experiment. Near the obstacle, the problem takes more than one iteration to converge, as the robot needs to reach its target while avoiding the obstacle. Additionally, the time taken by the memory warm-start refinement rises significantly near the obstacle, taking up to 0.1 second to produce an efficient warm-start. As the computation is done in parallel to the MPC, increased memory time simply implies that the MPC will remain stuck for a while in a local minimum.

Fig. 8 shows how the robot avoids the obstacle by letting its arm down and going up again. The obstacle (a 2 cm diameter vertical pole) is put at coordinates $(x = 0.6, y = 0.3)$. As shown in Fig. 7, the memory switch implies some control discontinuity, corresponding to a warm-start which is very dissimilar to the previous solution. The discontinuity could be smoothed by extending the preview window or by enforcing stronger continuity constraints on the torque trajectory.

VI. Conclusion

In this paper, whole body MPC in real time has been proved to be feasible at a frequency rate of

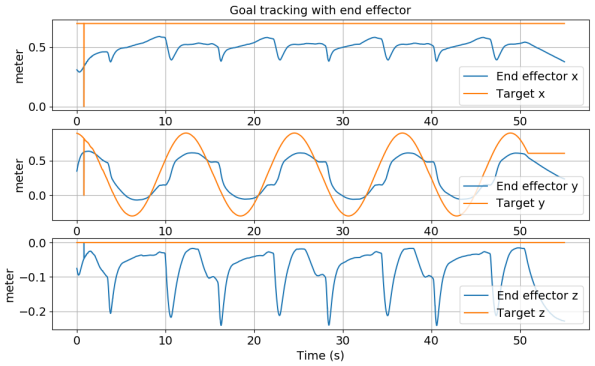

Fig. 8: Tracking of a linear moving target with the left arm end-effector in obstacle experiment.

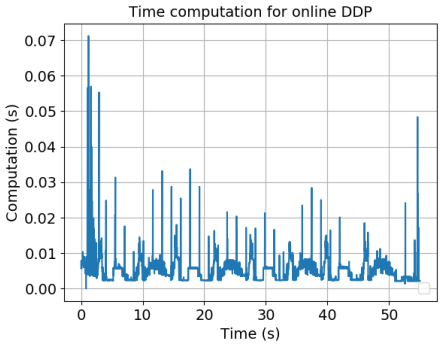

Fig. 9: Time computation of at most 5 DDP iterations for the obstacle experiment. Mean value is 4.98 ms.

100 Hz. This MPC formulation is based on a DDP implementation with augmented KKT dynamic which allows to efficiently compute a future trajectory of 0.5 second for a 22 degree of freedom robot model. Our two-step implementation, beginning with an offline DDP computation until convergence and followed by an online refinement over one iteration, produces a proper stable torque control able to reject external perturbations and to handle various tasks, from center of mass tracking to posture regularization.

Additionally, this paper introduces a strategy to incorporate an efficient DDP initialization scheme based on a memory of motion to the MPC implementation. This scheme has been successfully implemented on the robot to solve non-convex problems at a 100 Hz frequency. Collision avoidance in real time has been performed on a reduced model for a reaching task. In simulation, we also observed that the MPC without memory cannot pass the obstacle and instead remains stranded at local optima, waiting for the target to return: this shows how the memory warm start is essential to solving non-convex problems in path planning.

This is the first time DDP is used to control a complete humanoid with state feedback. This first study concentrated on analysing the behavior of our controller, which resulted in good performance. The proposed tasks remain simple benchmarks, but they pave the road for more elaborated tasks. Our next step is to control the locomotion of Talos using the proposed approach.

REFERENCES

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

[2] M. Huba, S. Skogestad, M. Fikar, M. Hovd, T. A. Johansen, and B. Rohal-Ilkiv, "Selected topics on constrained and nonlinear controls," *STU Bratislava – NTNU Trondheim*, 2011.

[3] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[4] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[5] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control," *19th Mediterranean Conference on Control Automation (MED)*, p. 1247–1252, 2011.

[6] E. Fernandez-Camacho and C. Bordons-Alba, *Model Predictive Control in the Process Industry*. Springer, London, 1995.

[7] J. Rawlings, E. Meadows, and K. Muske, "Nonlinear model predictive control: A tutorial and survey," *IFAC Advanced Control of Chemical Processes*, 1994.

[8] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," *21st Benelux Meeting on Systems and Control*, 2020.

[9] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[10] 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 international conference on robotics and automation (ICRA)*, 2016.

[11] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 2958–2964.

[12] B. Houska and M. Diehl, "Robustness and stability optimization of power generating kite systems in a periodic pumping mode," in *2010 IEEE International Conference on Control Applications*. IEEE, 2010, pp. 2172–2177.

[13] 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," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[14] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," *arXiv preprint arXiv:1909.06586*, 2019.

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

[16] 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," *System Integration (SII) 2019 IEEE/SICE International Symposium*, 2019.

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

[18] D. Pardo, L. Möller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating direct transcription and nonlinear optimization methods for robot motion planning," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 946 – 953, 2016.

[19] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guidance*, vol. 10, no. 4, p. 338–342, 2016.

[20] 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.

[21] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," *1st International Conference on Informatics in Control, Automation and Robotics*, 2004.

[22] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[23] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[24] A. Majumdar, G. Hall, and A. A. Ahmadi, "Recent scalability improvements for semidefinite programming with applications in machine learning, control, and robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 331–360, 2020.

[25] 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.

[26] N. Mansard, A. D. Prete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[27] T. S. Lembono, C. Mastalli, P. Fernbach, N. Mansard, and S. Calinon, "Learning how to walk: Warm-starting optimal control solver with memory of motion," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[28] 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 International Conference on Humanoid Robots*, 2014.

[29] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *International Conference on Intelligent Robots and Systems*, 2014.

[30] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. D. Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "Talos: A new humanoid research platform targeted for industrial applications," *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017.

[31] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018.

[32] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[33] 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.

[34] R. Bellman, "The theory of dynamics programming," *the Rand Corporation*, 1954.

[35] 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," *International Conference on Robotics and Automation*, 2020.

[36] 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.

[37] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175, 2014.

[38] T. S. Lembono, A. Paolillo, E. Pignat, and S. Calinon, "Memory of motion for warm-starting trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2594–2601, 2020.

[39] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[40] J. Mirabel, S. Tonneau, P. Fernbach, A. Seppälä, M. Campana, N. Mansard, , and F. Lamiraux, "Hpp: A new software for constrained motion planning," *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[41] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM 18*, 1975.