Whole-Body MPC without Foot References for the Locomotion of an Impedance-Controlled Robot

Alessandro Assirelli, Fanny Risbourg, Gianni Lunardi, Thomas Flayols, Nicolas Mansard

Whole-Body MPC without Foot References for the Locomotion of an Impedance-Controlled Robot

Alessandro Assirelli[1], Fanny Risbourg[1], Gianni Lunardi[2], Thomas Flayols[1,3], Nicolas Mansard[1,3]

Abstract— **With the fast progress of quadruped robots, we also see the rise of advanced controllers able to take whole-body decisions without any model reduction. Recently, whole-body model predictive control have been demonstrated on several legged robots. Based on these results, this paper presents a novel walking controller. Contrary to previously demonstrated approaches, our controller does not require the pre-computation of guide trajectories for the foot, nor of specific foot location, but rather decides on the flight the best foot movement using an original cost formulation. The predictive controller is then applied at the actuator level using an impedance controller, without requiring the more costly low-level torque controller that previous methods used. The method is validated on the real robot Solo, using an open-source implementation based on the solver Crocoddyl. We evaluate in depth the quality of the produced walk, despite external disturbance, and provide longer experiments in the companion video.**

I. Introduction

As more quadruped robots are built over the world, and impressive videos show dynamic motions in a wide range of environments, the problem of quadruped locomotion also becomes more trending, with a wide variety of approaches experimented. Hardware and computation times improvements allow the exploration of new frameworks and tools. Among them, Optimal Control Problems (OCP) are popular to formulate the robot controllers by scoring, using tailored objective functions and a prediction of the future behaviour of the robot. The definition of objectives and constraints is adapted to locomotion problems. OCPs can be used for trajectory optimization or for guiding the exploration process of Reinforcement Learning algorithms [2]. Trajectory optimization was once reserved for offline computation but can now be used online in real-time thanks to Model Predictive Control (MPC) frameworks [3]–[5]. MPC consists in successively solving finite-time OCPs over a receding horizon. The OCPs are typically initialized with a current estimate of the robot state.

The great number of variables that come from the discretization of the time horizon and the number of degrees of freedom of the system leads to challenges in meeting the computation time objectives. Thus MPC has been firstly used on systems with few degrees of freedom [6]–[8]. For quadruped robots which usually have twelve motors, reduced-models have been widely studied, but require the use of an additional whole-body controller to compute joint controls [9], [10]. They demonstrated the ability to plan robust motions in constrained environments while optimizing contact timings or locations [11].

To further simplify the OCP some pre-computations are typically done, such as the flying feet trajectories. They can be simple polynomials [12], or require more advanced computations to take into account obstacles on the way [13]. In any case, it is typically given as an input to the OCP in the form of a foot position tracking cost [14]. Different approaches [15], [16] have been studied, giving to the controller the freedom to choose the feet trajectories, at the cost of slower computations. In [17], the whole-body solver is guided by a centroidal trajectory [18], optimized in alternance to the kinematic trajectory. By only using simple cost guidance, the author shown that the whole-body solver is able to produce nice foot movements. In this paper, we propose a more complete way to describe the foot movement by an effective cost for the OCP, which enables the solver to find on its own the swing feet trajectories.

Recently, whole-body MPCs have been developed for legged robots [19]–[21]. The choice of Optimal Control numerical solver is decisive to reach the computation time limits. The OCP computes joint torque and state trajectories over an horizon, but can still not be solved at a frequency high enough to match the low-level controller of the robots [22]. To alleviate this limitation, Riccatti gains [23] can be used to adjust the torques sent to the servo controllers of the robot. For robots with torque feedback it is sufficient. Other robots, such as the Solo quadruped [24], require impedance control to efficiently control the joints. In this case, joint state targets are sent as well as the feed-forward torques to the low-level controller.

How can whole-body MPC be implemented on real robots without torque feedback ? This paper proposes a solution that was experimentally validated on the Solo quadruped robot. It presents two methods to compute the joint states necessary to impedance control. The first one is a spline interpolation of the discretized state computed by the OCP. The second one is the integration of the torques computed with the Riccatti gains. The two methods are tested in simulation and their performances are compared. The interpolation is used for experiments on the robot.

Firstly, the underlying concepts and mathematical theories are presented in Sec. II. Secondly in Sec. III the computation of low-level targets for an impedance based robot, first contribution of the paper is detailed. In Sec. IV, the focus is made on the OCP formulation with an emphasis on the

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2]Industrial Engineering Department, University of Trento
[3]Artificial and Natural Intelligence Toulouse Institute, France

An extended video and the software implementing the proposed method are available on the project web page [1].

second contribution, the flying foot cost design. Finally the experimental evaluation and results are presented in Sec. V.

II. Whole-Body control for Legged Locomotion: The Theory Behind It

In this section the OCP framework used to implement our locomotion controller is briefly presented, see [19] for details with stiff contacts.

A. Rigid-Body dynamics

The robot is described as a rigid-body system. Thus the dynamics with its environment is the following [25]:

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

where $\mathbf{q} \in SE(3) \times \mathbb{R}^{n_j}$ is the configuration vector of the robot, containing the free-flyer position (base of the robot for example) and the n_j joint positions; $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are respectively the velocity and acceleration vectors of the robot, laying in the tangent space of $SE(3) \times \mathbb{R}^{n_j}$; τ is the torque vector and S a selection matrix for the actuated joints; λ is the concatenation of the contact forces and $\mathbf{J}_c$ the concatenation on the contact Jacobians; $\mathbf{M}$ is the inertia matrix of the robot and $\mathbf{b}$ gathers the generalized non-linear forces. In the considered case of quadruped robots, the contacts are punctual and the forces λ are modeled as 3D forces constrained to lie in a friction cone.

The state of the robot is $\mathbf{x} = (\mathbf{q}, \dot{\mathbf{q}})$ and the control $\mathbf{u} = \tau$. From (1) we deduce that the contact forces can be computed from the state and control and hence do not need to be variables of the problem. The dynamics $\dot{\mathbf{x}}$ can be deduces as a function $f(\cdot)$ of $\mathbf{x}$ and $\mathbf{u}$:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \tag{2}$$

B. Optimal Control

Our OCP is formulated as an optimality problem on a time horizon where the variables are trajectories for state and control over the horizon:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \int_0^T l(\mathbf{x}(t), \mathbf{u}(t), t) + l_T(\mathbf{x}(T)) \\ s.t. \quad & e(\mathbf{x}(0), \mathbf{u}(0), \mathbf{x}(T), \mathbf{u}(T)) = 0 \\ & \forall\, t \in [0, T]: \\ & \quad \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), t) \\ & \quad h(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0 \end{aligned} \tag{3}$$

where $\mathbf{x} : t \mapsto \mathbf{x}(t)$ and $\mathbf{u} : t \mapsto \mathbf{u}(t)$ are respectively the state and control, both functions of time t; $h(\cdot)$ encompasses the paths constraints and $e(\cdot)$ the endpoint conditions.

A common approach to solve an OCP is to rely on *direct methods* [26], which discretize the problem to obtain a nonlinear optimization problem that is a finite-dimensional approximation of the original OCP. First, time is discretized in $n+1$ so-called *nodes*: $[t_0, ..., t_n]$; then $\mathbf{X}$ and $\mathbf{U}$ are discretizations of $\mathbf{x}$ and $\mathbf{u}$ at each node:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(t_0) \triangleq \mathbf{x}_0 \\ \mathbf{x}(t_1) \triangleq \mathbf{x}_1 \\ \vdots \\ \mathbf{x}(t_n) \triangleq \mathbf{x}_n \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}(t_0) \triangleq \mathbf{u}_0 \\ \mathbf{u}(t_1) \triangleq \mathbf{u}_1 \\ \vdots \\ \mathbf{u}(t_{n-1}) \triangleq \mathbf{u}_{n-1} \end{bmatrix} \tag{4}$$

The formulation of the discretized problem is:

$$\begin{aligned} \min_{\mathbf{X},\mathbf{U}} \quad & \sum_{t=0}^{n-1} \ell_t(\mathbf{x}_t, \mathbf{u}_t) + \ell_T(\mathbf{x}_n) \\ s.t. \quad & e(\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_n, \mathbf{u}_{n-1}) = 0 \\ & \forall\, t \in [\![0, n-1]\!]: \\ & \quad \mathbf{x}_{t+1} = f_t(\mathbf{x}_t, \mathbf{u}_t) \\ & \quad h_t(\mathbf{x}_t, \mathbf{u}_t) \leq 0 \end{aligned} \tag{5}$$

where $f_t(\cdot)$ is a discretized version of the continuous-time dynamics obtained by numerical integration [27] (e.g., Runge-Kutta methods).

C. Feasibility-driven Differential Dynamic Programming

A recent solver for multiple-shooting OCP is the Feasibility-driven Differential Dynamic Programming (FDDP) [20] algorithm. Given the difference between the new and actual values

$$(\delta\mathbf{x}_t, \delta\mathbf{u}_t) = (\hat{\mathbf{x}}_t - \mathbf{x}_t, \hat{\mathbf{u}}_t - \mathbf{u}_t), \tag{6}$$

the Value function V_t is locally approximated as:

$$V_t(\delta\mathbf{x}_t) = \min_{\delta\mathbf{u}_t} \ \ell(\delta\mathbf{x}_t, \delta\mathbf{u}_t) + V_{t+1}(f_t(\delta\mathbf{x}_t, \delta\mathbf{u}_t)) \tag{7}$$

With respect to the traditional DDP [28], FDDP introduces intermediate shooting nodes as further decision variables. The objective is to close the gaps (also called defects [29]) $\bar{f}_{t+1}$ between the rollout of the dynamics and the state trajectory:

$$\bar{f}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{x}_{t+1} \tag{8}$$

The local approximation of the Hamiltonian $\mathbf{Q}$ writes:

$$\begin{aligned} \mathbf{Q}_{\mathbf{x}_t} &= \boldsymbol{\ell}_{\mathbf{x}_t} + f_{\mathbf{x}_t}^T \mathbf{V}_{\mathbf{x}_{t+1}}^+ \\ \mathbf{Q}_{\mathbf{u}_t} &= \boldsymbol{\ell}_{\mathbf{u}_t} + f_{\mathbf{u}_t}^T \mathbf{V}_{\mathbf{x}_{t+1}}^+ \\ \mathbf{Q}_{\mathbf{xx}_t} &= \boldsymbol{\ell}_{\mathbf{xx}_t} + f_{\mathbf{x}_t}^T \mathbf{V}_{\mathbf{xx}_{t+1}} f_{\mathbf{x}_t} \\ \mathbf{Q}_{\mathbf{xu}_t} &= \boldsymbol{\ell}_{\mathbf{xu}_t} + f_{\mathbf{x}_t}^T \mathbf{V}_{\mathbf{xx}_{t+1}} f_{\mathbf{u}_t} \\ \mathbf{Q}_{\mathbf{uu}_t} &= \boldsymbol{\ell}_{\mathbf{uu}_t} + f_{\mathbf{u}_t}^T \mathbf{V}_{\mathbf{xx}_{t+1}} f_{\mathbf{u}_t} \end{aligned} \tag{9}$$

where $\boldsymbol{\ell}_{\mathbf{x}_t}$, $\boldsymbol{\ell}_{\mathbf{u}_t}$ and $\boldsymbol{\ell}_{\mathbf{xx}_t}$, $\boldsymbol{\ell}_{\mathbf{xu}_t}$, $\boldsymbol{\ell}_{\mathbf{uu}_t}$ are the Linear Quadratic approximations of the cost function; $f_{\mathbf{x}_t}$, $f_{\mathbf{u}_t}$ are the Jacobians of the dynamics. $\mathbf{V}_{\mathbf{x}_{t+1}}$ and $\mathbf{V}_{\mathbf{xx}_{t+1}}$ are respectively the gradient and the Hessian of the Value function, with $\mathbf{V}_{\mathbf{x}_{t+1}}^+ = \mathbf{V}_{\mathbf{x}_{t+1}} + \mathbf{V}_{\mathbf{xx}_{t+1}} \bar{f}_{t+1}$ being the gradient after the deflection due to the gap $\bar{f}_{t+1}$. The new trajectory $\mathbf{x}$ is achieved through the integration of the dynamics with the forward pass:

$$\begin{aligned} \hat{\mathbf{u}}_t &= \mathbf{u}_t + \alpha\mathbf{k}_t + \mathbf{K}_t(\hat{\mathbf{x}}_t - \mathbf{x}_t) \\ \hat{\mathbf{x}}_{t+1} &= f_t(\hat{\mathbf{x}}_t, \hat{\mathbf{u}}_t) - (1-\alpha)\bar{f}_{t+1} \end{aligned} \tag{10}$$

where $\alpha \in [0, 1]$ is the step of the line search, while $\mathbf{k}_t$ and

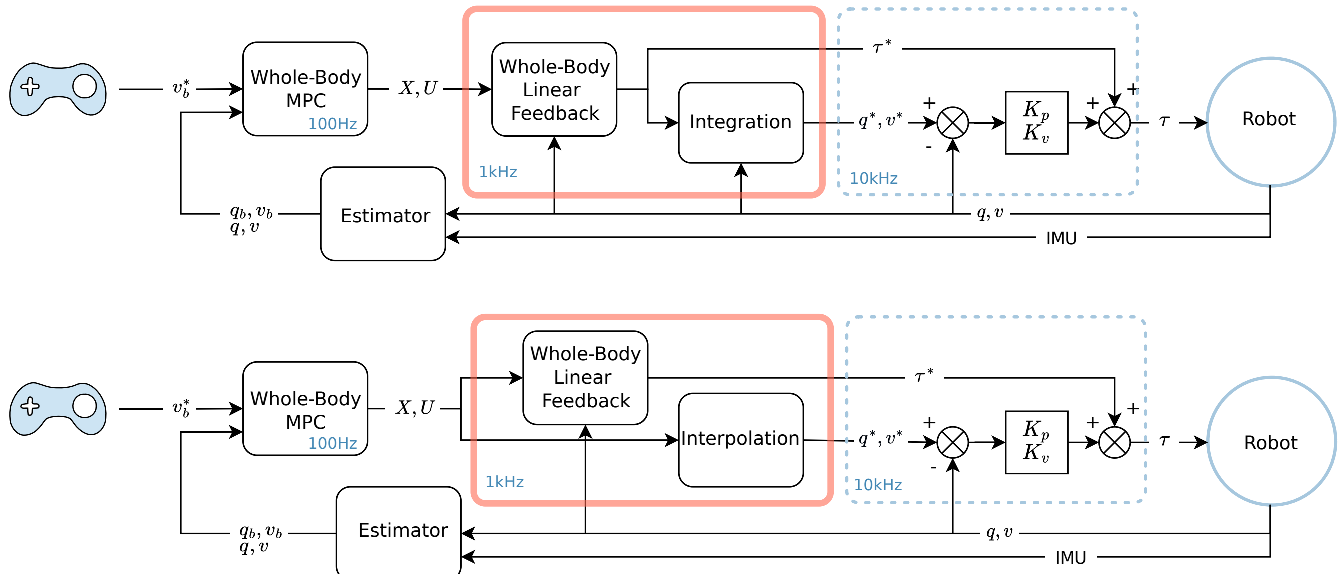

Fig. 1. Solo walking controller architecture, using the integration (Sec. III-C.1 (top) or the interpolation (Sec. III-C.2 (bottom).

$\mathbf{K}_t$ are respectively the feed-forward and feedback (Riccatti) gains computed from the backward pass (9):

$$\mathbf{k}_t = \mathbf{Q}_{\mathbf{uu}_t}^{-1}\mathbf{Q}_{\mathbf{u}_t}, \quad \mathbf{K}_t = \mathbf{Q}_{\mathbf{uu}_t}^{-1}\mathbf{Q}_{\mathbf{xu}_t} \tag{11}$$

We check the Goldstein condition to accept the trial step length. For the quadruped locomotion problems used in the experiments, the solver can iterate at 100HZ.

D. Riccati gains

The optimal control policy of (5) computed by the DDP can be expressed as a function π of the state, whose derivative is the Riccati gain [23]:

$$\mathbf{u}_t = \pi_t(\mathbf{x}), \quad \left.\frac{\partial \pi_t}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_t} = \mathbf{K}_t \tag{12}$$

These gains will be used to compute the optimal torques to apply, using a linear feedback on the state, as detailed in Sec. III-B.

III. Impedance targets computation

Most OCP for locomotion are written, as ours, at torque level. Following, the previously proposed whole-body MPC is built assuming the robot is able to properly track the reference torques $\mathbf{u}_0 = \pi\mathbf{x}$. Yet only few robots are equipped with torque sensors and low-level torque feedback. Rather, like the robot Solo used for the experiments of this paper, the actuators are driven by impedance, tracking a given reference state. In this section, we propose a method to bridge the whole-body MPC to the low-level impedance control.

A. Actuator feedback

The robot is controlled by impedance, in the form of a Proportional-Derivative (PD) plus feed-forward controller. The target joint torques τ^*, positions $\mathbf{q}^*$ and velocities $\dot{\mathbf{q}}^*$ are sent to the motor control board every millisecond. The boards compute the torques to apply using joint positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$ measured by the encoders:

$$\tau = \tau^* + K_p(\mathbf{q}^* - \mathbf{q}) + K_d(\dot{\mathbf{q}}^* - \dot{\mathbf{q}}) \tag{13}$$

The impedance loop (13) is embedded and cycles at 10kHz. Impedance targets τ^*, $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$ are updated at 1kHz. Recall that the whole-body MPC iterates at 100Hz. We now describe the proposed mehods to compute the reference τ^*, $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$, summarized in Fig 1.

B. Whole-Body Linear Feedback for Reference Torques Computation

The OCP provides discretized reference joint torque trajectories. The discretization step is 12ms. To compute the reference torques to send at 1kHz, whole-body linear feedback is realized using the Riccatti gains $\mathbf{K}_i$:

$$\tau^* = \tau_0 + \mathbf{K}_0(\mathbf{x} - \mathbf{x}_0) \tag{14}$$

where $\tau_0 = \pi_0(\mathbf{x}_0)$ is the initial reference torque computed by the OCP, $\mathbf{x}_0$ the initial state given to the OCP at 100Hz, $\mathbf{x}$ is the measured state estimated at 1kHz.

C. Joint state targets re-sampling for impedance control

There are several ways to compute the joint state targets sent to the low-level controller in (13). Two possibilities are presented in the following sections.

1) Integration: The first one is the integration of the reference torques. The acceleration $\mathbf{a}$ is obtained by computing the forward dynamics (implemented in the Pinocchio library [30] using the Articulated Body Algorithm, ABA), from the current joint state measurements, and reference torques τ^* in (14).

$$\mathbf{a} = ABA(\mathbf{q}, \dot{\mathbf{q}}, \tau^*, \lambda) \tag{15}$$

$$\dot{\mathbf{q}}^* = \dot{\mathbf{q}} + \mathbf{a} \cdot dt \tag{16}$$

$$\mathbf{q}^* = \mathbf{q} + \dot{\mathbf{q}}^* \cdot dt \tag{17}$$

where λ is a by product of the OCP solver following (1) (and is void when the robot runs without contact).

2) Interpolation: The second proposition is to chose $\mathbf{x}$ as a polynomial interpolation of the discretized state trajectory $\mathbf{X}$ computed by the OCP. The Krogh interpolator [31] computes an interpolation of the trajectory points and their associated derivatives. The first three samples $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ of the state trajectory are used to constrain the interpolation.

IV. Optimal Control Problem Formulation for Locomotion

This section presents the definition of the OCP used in our locomotion controller. It is implemented with Crocoddyl [20]. The dynamics equations have been presented in Section II. The contact timings are predefined and passed along to define the contact constraints used for each node of the OCP. To simplify the notation of cost weights, they will appear as:

$$\|\xi\|_w^2 = \sum_{i=1}^{n} w_i \xi_i^2, \quad \mathbf{w} = (w_1, \dots, w_n) \tag{18}$$

where the weights w will often be reduced to a scalar.

The running cost $\ell_t(\mathbf{x}, \mathbf{u})$ is defined as the sum of the following costs.

A. Flying foot cost

In typical locomotion problem [14], [20], a quadruped follows predefined swing feet trajectories. In this work we substitute the foot desired trajectory with the following cost model:

$$\ell_{fly}^f(\mathbf{x}) = \frac{1}{\mathrm{e}^{\gamma \mathrm{h^f}}} \left\|\mathbf{v}_{xy}^f\right\|_{w_{fly}}^2 \tag{19}$$

where f is a foot that is not in contact; $\mathbf{v}_{xy}^f = (v_x^f, v_y^f) \in \mathbb{R}^2$ contains the longitudinal and lateral components of the foot linear velocity; h^f is the height of the foot; $w_{fly} = 5 \cdot 10^4$, $\gamma = 50$ are hyper-parameters. The solver will seek to raise the altitude when high velocity $\mathbf{v}^f$ is needed, while keeping the velocity small close to the ground. We will show in the experiments that this cost produces bell-shaped foot trajectories without needing (as done in previous whole-body OCPs) to impose a reference trajectory or even contact locations.

B. Impact costs

Two costs are used to handle properly the impact of a foot on the ground. They are activated only when a foot f changes from a no contact to a contact phase. The first one imposes the height of the foot at contact h^f to be the height h_0 of the foot at initialization. The second one imposes the linear velocity of the foot $\mathbf{v}^f \in \mathbb{R}^3$ to be 0 when it creates the contact.

$$\begin{aligned} \ell_h^f(\mathbf{x}) &= \left\|h^f - h_0\right\|_{w_h}, \qquad & w_h &= 10^4 \\ \ell_i^f(\mathbf{x}) &= \left\|\mathbf{v}^f\right\|_{w_i}, \qquad & w_i &= 10^4 \end{aligned} \tag{20}$$

C. Regularization

Three costs are used to regularize (thus limit) the state, control and contact forces:

$$\begin{aligned} \ell_x(\mathbf{x}) &= \|(\mathbf{x} - \mathbf{x}^*)\|_{w_x}^2, \qquad & w_x &= [\mathbf{0}_7 \ \mathbf{3} \cdot 10_{12}^2 \ \mathbf{0}_6 \ \mathbf{20}_{12}]^T \\ \ell_u(\mathbf{u}) &= \|\mathbf{u}\|_{w_u}^2, \qquad & w_u &= 10^4 \\ \ell_\lambda(\mathbf{x}, \mathbf{u}) &= \|\lambda - \lambda^*\|_{w_\lambda}^2, \qquad & w_\lambda &= 10^2 \end{aligned} \tag{21}$$

where $\mathbf{x}^* = [\mathbf{q}_0 \ \mathbf{0}_{n_v}]^T$ with $\mathbf{q}_0$ the initial configuration of the robot. And for the feet in contact the reference force is a fraction of the robot weight:

$$\lambda^* = \left[0, 0, \frac{mg}{n_{cts}}\right]^T$$

with n_{cts} the number of feet in contact. otherwise $\lambda^* = \mathbf{0}_3$.

The joint velocity is also regularized with a terminal cost:

$$\ell_T(\mathbf{x}) = \|(\mathbf{x}_T)\|_{w_{term}}^2, \quad w_{term} = [\mathbf{0}_{n_q} \ \mathbf{10}_{n_v}^3]^T \tag{22}$$

with n_q the dimension of the state position.

D. Boundaries

FDDP cannot enforce any other constraint than (1). Boundaries on the joint velocities and control, are implemented with the following penalties:

$$\begin{aligned} \ell_{bv}(\mathbf{x}) &= \|\min(\max(\mathbf{x}, \bar{\mathbf{x}}), \underline{\mathbf{x}})\|_{w_{b_x}}^2, \quad & w_{b_x} &= [\mathbf{0}_{n_q+6} \ \mathbf{10}_{n_v-6}]^T \\ \ell_{bu}(\mathbf{u}) &= \|\min(\max(\mathbf{u}, \bar{\mathbf{u}}), \underline{\mathbf{u}})\|_{w_{b_u}}^2, \quad & w_{b_u} &= 10^4 \end{aligned} \tag{23}$$

where $\bar{\mathbf{x}}$, $\underline{\mathbf{x}}$, $\bar{\mathbf{u}}$ and $\underline{\mathbf{u}}$ are the velocity and torque boundaries adapted to the robot.

An addition boundary cost was added on the height of each flying foot f to prevent the OCP to select solutions where a flying foot would penetrate the ground:

$$\ell_g(\mathbf{x}) = \left\|\max(h^f, h_0)\right\|_{w_g}^2, \quad w_g = 10^3 \tag{24}$$

E. Reference base velocity tracking

Finally, the robot velocity is tracked using a dedicated cost:

$$\ell_t(\mathbf{x}) = \|\mathbf{v}_{base} - \mathbf{v}_{base}^*\|_{w_t}^2, \quad w_v = 8 \cdot 10^5 \tag{25}$$

in which $\mathbf{v}_{base} \in \mathbb{R}^6$ is the spatial velocity of the robot's base and $\mathbf{v}_{base}^* \in \mathbb{R}^6$ a reference velocity obtained from a gamepad for example (see Fig 1).

V. Experimental Results

Experiments were realized to compare some design choices and assess the performances of the controller. They were realized with the Solo quadruped robot.

A. The Solo quadruped

1) Hardware: Solo [24] is a 4-legged robot, developed by the Open Dynamic Robot Initiative. It is a light and small (22 cm high for 2.5 kg) robot, rather cheap and easy to maintain thanks to its 3D-printed parts and off-the-shelves components. Its 12 actuated degrees of freedom are equipped with brushless outrunner motors and low ratio timing belt gearbox, adapted to open loop torque control. The sensors mounted on the robot are 12 encoders located at each joint and an IMU embedding an extended Kalman filter [12].

2) Controller framework: The control loop runs on a distant computer (Intel(R) Core(TM) i5-9500 CPU @ 3.00GHz) to compute the impedance targets τ^*, $\mathbf{q}^*$ and $\dot{\mathbf{q}}^*$. Simulation tests were realized using the PyBullet [32] physics engine.

The controller framework is shown in Fig. 1. A gamepad is used to decide the reference velocity of the base of the robot (x, y and yaw), expressed in the local frame. The MPC computes at 100Hz the state and torques over an horizon of 480ms. The joint states and torque impedance targets are then computed at 1kHz and sent to the control boards of the robot. The estimator provides an estimation of the base position and velocity of the robot.

3) Base velocity estimator: The estimator combines forward kinematics data and IMU measurements to estimate the base 6d velocity.

The inertial measurement unit (IMU) used on the Solo quadruped includes an extended Kalman filter (EKF) that estimates the body angular velocity, orientation and linear acceleration unbiased from gravity. Leg kinematic is measured by high resolution optical encoder. Joint velocity is obtained by finite differentiation.

The base angular velocity is directly measured by the IMU's gyroscopes. The base linear velocity however needs to be estimated by sensor fusion. For this, we use a simple linear complementary filter as described in [12] and briefly summarised below.

a) Low frequency source: For each foot in contact, assuming a rigid point contact without slipping, the leg kinematic can give a partial information about the base velocity with respect to the contact point. Using base orientation and angular velocity estimates from IMU's embeded EKF, we can fully reconstruct the base velocity. Once averaged over each contact point, this data source give a noisy but un-biased estimate.

b) High frequency source: Since the IMU's EKF provide linear acceleration with gravity removed, we can integrate this quantity to estimate base velocity. This will capture the rapid changes in the velocity, but suffer from integration drift, making it a good high frequency source.

The complementary filter combines this two data sources as the sum of 1st order low pass filter estimation from the kinematics, and 1st order high pass filter of the IMU integration.

For all experiment, the cutoff filter frequencies are set to 5Hz.

B. Joint states targets comparison

1) Reduced model experiments: To investigate how the re-sampling of the state trajectories impacts the behaviour of the robot (see Section III), we designed an OCP that can run at 1kHz, using a reduced model of the robot, with only three degrees of freedom. The objective of the MPC is to track a foot position trajectory (sinus in y and z). It enabled us to compare the tracking performance of the MPC evaluated at 1kHz using directly the targets computed by the OCP, with the same MPC evaluated at 100Hz using Riccatti gains

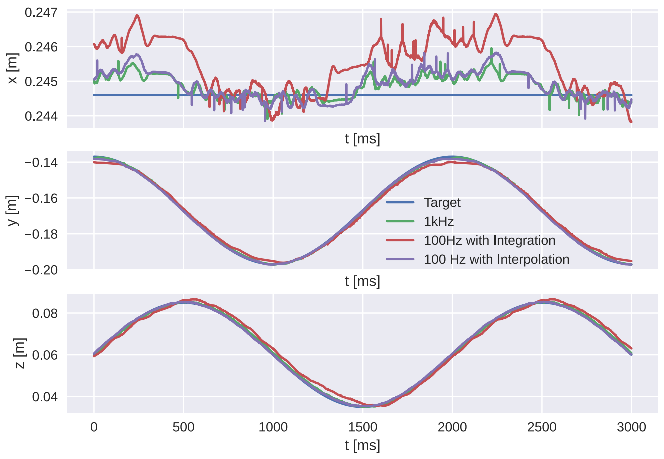

Fig. 2. Experiment V-B.1. Foot tracking comparison in 3DOF experiment

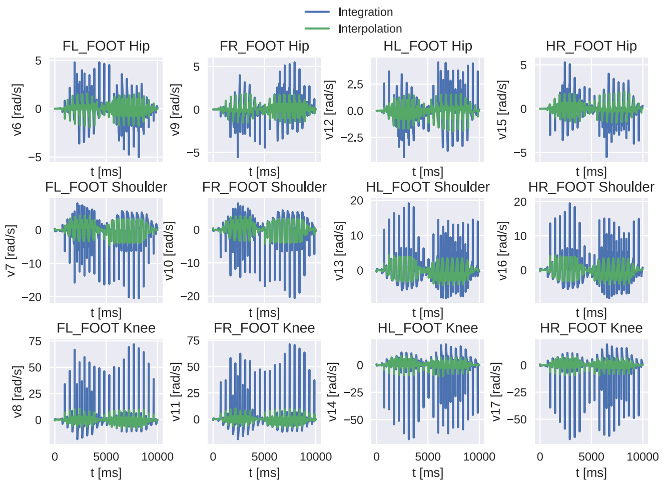

Fig. 3. Experiment V-B.2. Joint velocity references

and re-sampling. The torques and joint states were computed using Riccatti gains and either interpolation or integration, as described in part V-A.2. Fig. 2 shows the foot position trajectories of these three controllers during an experiment on the real robot. We can see that they are rather similar. The impact of running the MPC at a lower frequency and using Riccatti gains and re-sampling did not degrade the behaviour of the robot. Integration shows a slightly worse tracking than interpolation.

2) Locomotion experiments in simulation: To further study the differences between integration and interpolation, we performed locomotion experiments. On the robot, the integration framework did not work, so we realized the experiments in simulation. We compared the interpolation III-C.2 and integration III-C.1 schemes to compute the target joint states between two nodes of the OCP. Fig. 3 shows the joint velocity targets computed by each mean, in a simulation with no noise. We can see that the interpolated velocities are smoother than the integrated, which have bigger peaks. The velocity tracking of the base of the robot looks slightly more efficient for the interpolation, as can be seen in figure 4. Moreover, when we add noise in the simulation (in the

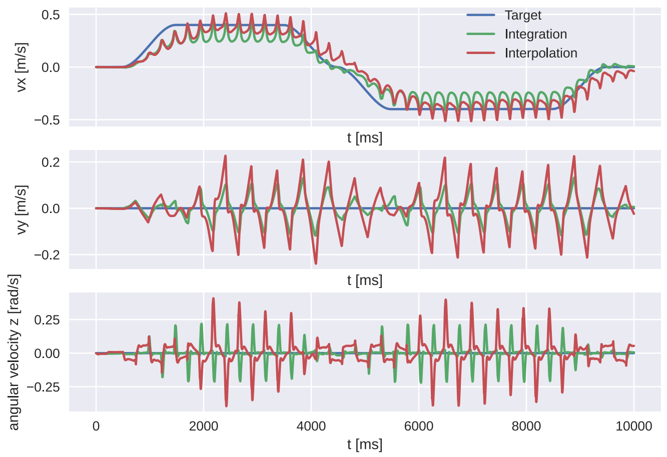

Fig. 4. Experiment V-B.2. Base velocity tracking comparison between interpolation and integration

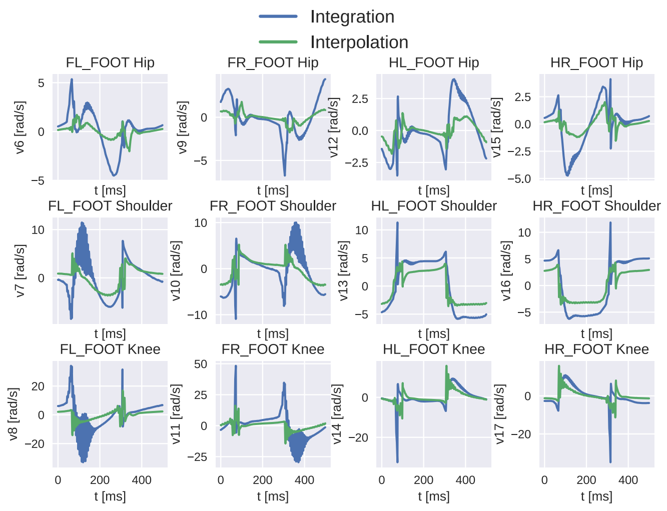

Fig. 5. Experiment V-B.2. Joint velocity references in a simulation with noise

torques applied to the joints and the measured data), we observe that the integrated velocity target become noisier, as can be show in figure 5.

These experiments lead us to select the interpolation framework for the locomotion experiments on the robot.

C. Walking

The walking experiments conducted on the robot demonstrated the robustness of the controller. The robot was able to follow linear velocity references up to 0.5m/s, as well as angular ones up to 1rad/s. Push recovery tests were successfully experimented, with the controller able to recover from an impact, and adapting to a constant force pushing it. The companion video shows some footage of these experiments.

Fig. 6 shows the base velocity tracking performance. The foot clearance can be seen in Fig. 7. We can observe that when the linear velocity of the base increases, the swing foot moves higher.

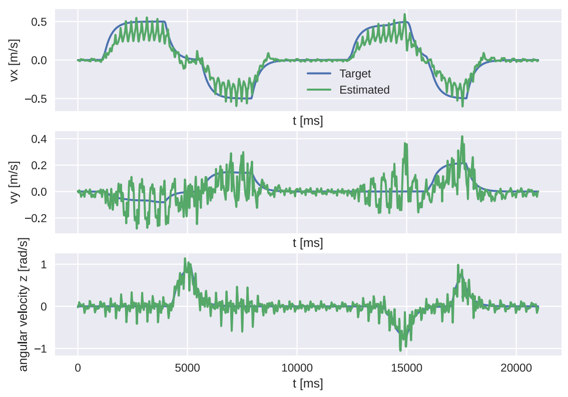

Fig. 6. Experiment V-C. Base velocity tracking during walking test

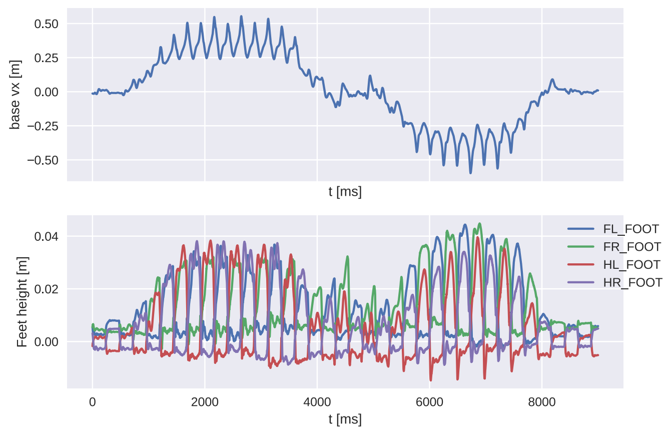

Fig. 7. Experiment V-C. Base velocity and foot clearance during locomotion

VI. Conclusion and Future work

Through this paper we presented our work on a novel locomotion controller for legged robots, based on whole-body MPC. Traditional polynomial foot trajectory targets where replaced in the OCP by an innovative cost, hence letting full freedom to the solver to choose the best foot movement. We also proposed a generic solution to implement the MPC on a robot without low-level torque control, by computing the impedance references using polynomial interpolation. Experiments in simulation and on the Solo quadruped were realized to justify some of the design choices: with the reduced three degrees of freedom model, the interpolation of the joint states ensures no loss of performances with respect to an OCP running at higher frequencies. Finally the robustness of the controller was assessed during walking experiments with perturbations. The novel flying foot cost led to proper reactive foot motion, where higher base velocity implies bigger swing foot height, as expected.

Future works will provide improvements on the flying cost, such that it will include distance measurements between obstacles rather than only height form the ground. This feature will lead to walking experiments in more complex environments (e.g. with stairs).

References

[1] "Project page," https://gepettoweb.laas.fr/articles/assirelli_2022.html.

[2] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1–9. [Online]. Available: https://proceedings.mlr.press/v28/levine13.html

[3] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer science & business media, 2013.

[4] J. Rawlings, E. Meadows, and K. Muske, "Nonlinear model predictive control: A tutorial and survey," *IFAC Proceedings Volumes*, vol. 27, no. 2, pp. 185–197, 1994.

[5] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," 01 2002.

[6] 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.

[7] M. Geisert and N. Mansard, "Trajectory generation for quadrotor based systems using numerical optimal control," in *IEEE ICRA*, 2016.

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

[9] 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.

[10] T. Corbères, T. Flayols, P.-A. Léziart, R. Budhiraja, P. Souères, G. Saurel, and N. Mansard, "Comparison of predictive controllers for locomotion and balance recovery of quadruped robots," in *IEEE ICRA*, 2021.

[11] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *IEEE IROS*, 2010.

[12] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," in *IEEE ICRA*, 2021.

[13] F. Risbourg, T. Corberes, P.-A. Leziart, T. Flayols, N. Mansard, and S. Tonneau, "Real time footstep planning and control of the solo quadruped robot in 3d environments," in *IEEE IROS*, 2022.

[14] E. L. Dantec, M. Naveau, N. Mansard, P. Fernbach, N. Villa, G. Saurel, O. Stasse, and M. Taïx, "Whole-Body Model Predictive Control for Biped Locomotion on a Torque-Controlled Humanoid Robot," Jul. 2022, working paper or preprint. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03724019

[15] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE RA-L*, 2018.

[16] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, "Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations," 2020.

[17] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti, "Biconmp: A nonlinear model predictive control framework for whole body motion planning," 2022.

[18] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, 2018.

[19] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon *et al.*, "Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos," *IEEE ICRA*, 2021.

[20] 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," in *IEEE ICRA*, 2020.

[21] S. Katayama and T. Ohtsuka, "Whole-body model predictive control with rigid contacts via online switching time optimization," in *IEEE IROS*, 2022.

[22] P. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *IEEE Transactions on Robotics*, 2017.

[23] E. L. Dantec, M. Taix, and N. Mansard, "First Order Approximation of Model Predictive Control Solutions for High Frequency Feedback," in *IEEE ICRA*, 2022.

[24] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE RA-L*, 2020.

[25] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential dynamic programming for multi-phase rigid contact dynamics," in *IEEE Humanoids*, 2018.

[26] E. Trélat, "Optimal control and applications to aerospace: some results and challenges," *Journal of Optimization Theory and Applications*, vol. 154, no. 3, pp. 713–758, 2012.

[27] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. SIAM, 2010.

[28] D. Q. Mayne, "Differential dynamic programming–a unified approach to the optimization of dynamic systems," in *Control and dynamic systems*, 1973, vol. 10.

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

[30] 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," in *IEEE/SICE International Symposium on System Integration*, 2019.

[31] F. T. Krogh, "Efficient algorithms for polynomial interpolation and numerical differentiation," *Mathematics of Computation*, vol. 24, pp. 185–190, 1970.

[32] E. Coumans and Y. Bai, "PyBullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.