Inverse-Dynamics MPC via Nullspace Resolution

Citation for published version:

Digital Object Identifier (DOI):
10.1109/TRO.2023.3262186

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
IEEE Transactions on Robotics

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Inverse-Dynamics MPC via Nullspace Resolution

Carlos Mastalli† † Carlos Mastalli is with the School of Engineering and Physical Sciences, Heriot-Watt University, U.K. (e-mail: c.mastalli@hw.ac.uk). Saroj Prasad Chhatoi is with Centro di Ricerca “Enrico Piaggio”, Università di Pisa, Italy (e-mail: s.chhatoi@studenti.unipi.it). Thomas Corbères, Steve Tonneau and Sethu Vijayakumar are with the School of Informatics, University of Edinburgh, U.K. (e-mail: t.corberes@sms.ed.ac.uk; stonneau@exseed.ed.ac.uk; sethu.vijayakumar@ed.ac.uk).

Abstract—Optimal control (OC) using inverse dynamics provides numerical benefits such as coarse optimization, cheaper computation of derivatives, and a high convergence rate. However, to take advantage of these benefits in model predictive control (MPC) for legged robots, it is crucial to handle efficiently its large number of equality constraints. To accomplish this, we first (i) propose a novel approach to handling equality constraints based on nullspace parametrization. Our approach balances optimality, and both dynamics and equality-constraint feasibility appropriately, which increases the basin of attraction to high-quality local minima. To do so, we (ii) modify our feasibility-driven search by incorporating a merit function. Furthermore, we introduce (iii) a condensed formulation of inverse dynamics that considers arbitrary actuator models. We also propose (iv) a novel MPC based on inverse dynamics within a perceptive locomotion framework. Finally, we present (v) a theoretical comparison of optimal control with forward and inverse dynamics and evaluate both numerically. Our approach enables the first application of inverse-dynamics MPC on hardware, resulting in state-of-the-art dynamic climbing on the ANYmal robot. We benchmark it over a wide range of robotics problems and generate agile and complex maneuvers. We show the computational reduction of our nullspace resolution and condensed formulation (up to 47.3%). We provide evidence of the benefits of our approach by solving coarse optimization problems with a high convergence rate (up to 10 Hz of discretization). Our algorithm is publicly available inside CROCODYL.

Index Terms—model predictive control, inverse dynamics, nullspace parametrization, legged robots, agile maneuvers.

I. INTRODUCTION

MODEL predictive control (MPC) of rigid body systems is a powerful tool to synthesize robot motions and controls. It aims to achieve complex motor maneuvers in real time, as shown in Fig. 1, while formally considering its intrinsic properties [1]: nonholonomic, actuation limits, balance, kinematic range, etc. We can describe rigid body dynamics through their forward or inverse functions, and efficiently compute them via recursive algorithms [2]. Similarly, we can formulate optimal control problems for rigid body systems using their forward and inverse dynamics. However, recent works on MPC (e.g., [3], [4], [5]) are based only on forward dynamics, as they can be efficiently solved via differential dynamic programming (DDP) [6]. In contrast, model predictive control with inverse dynamics poses new challenges. It requires handling equality constraints efficiently and exploiting its temporal and functional structure. On the one hand, with temporal structure, we refer to the inherent Markovian dynamics commonly encountered in optimal control problems. On the other hand, with functional structure, we refer to the sparsity pattern defined by inverse dynamics itself.

When formulating an optimal control problem via inverse dynamics and direct transcription [8], we decouple the integrator and robot dynamics. This imposes two different equality constraints on the nonlinear program. Such a strategy distributes the nonlinearities of both constraints, which helps to improve the convergence rate and deal with coarse discretization and poor initialization. Moreover, the time complexity of computing the inverse dynamics is lower than the forward dynamics [2]. This also applies to recursive algorithms that compute their analytical derivatives [9]. Both are among the most expensive computations when solving a nonlinear optimal control problem. Finally, the control inputs of inverse-dynamics formulations are the generalized accelerations and contact forces, instead of the joint torques as in forward-dynamics settings. This allows us to compute feedback policies for the generalized accelerations and contact forces, which can be easily integrated into an instantaneous controller (e.g., [10], [11]) to further ensure dynamics, balance, actuation limits, etc. are respected at a higher control frequency. Having such ad-

Fig. 1. ANYmal climbs up a damaged staircase using our inverse-dynamics MPC via our nullspace parametrization. In the instance depicted in this figure, ANYmal is crossing a gap incurred by a missing tread. The gap dimension is 26 cm in length and 34 cm in height, which represents an inclination of 37° and around half of the ANYmal robot. The footprint plan shown in the bottom-left corner is computed online thanks to our perceptive locomotion pipeline – details of which are described in [7] To watch the video, click the picture or see https://youtu.be/Nhv8UVopPCI.
A. Contribution

This work presents an efficient method for solving optimal control problems with inverse dynamics, which enables the first application of inverse-dynamics MPC in legged robots. It relies on four technical contributions:

(i) an efficient method based on nullspace parametrization for handling equality constraints,
(ii) a feasibility-driven search and merit function approach that considers both dynamics and equality-constraint feasibility,
(iii) a condensed inverse-dynamics formulation that handles arbitrary actuation models, and
(iv) a novel feedback MPC based on inverse dynamics integrated into a perceptive locomotion pipeline.

Our novel optimal-control algorithm enables inverse-dynamics MPC in legged robots, resulting in state-of-the-art dynamic climbing on the ANYmal robot. It uses acceleration and contact force policies to increase execution accuracy, which forms an integral part of feedback MPC approaches. An implementation of our algorithm is publicly available inside CROCODYLYL. In the following, we present a brief theoretical description of optimal control with forward and inverse dynamics after introducing related work. This section aims to provide details that help us to understand the benefits and challenges of inverse-dynamics MPC.

II. RELATED WORK

Recently, there has been interest in solving optimal control (OC) problems with inverse dynamics. Some of the motivations are faster computation of inverse dynamics and their derivatives [2], [9], convergence within fewer iterations [13], [14], and coarse problem discretization [15]. Despite these properties, these methods are slow for MPC applications, as they do not exploit the temporal and functional structure efficiently. In addition, their computational complexity increases with respect to the number of contacts. These reasons might explain why recent predictive controllers for legged robots are based only on forward dynamics [3], [4], [5], [12]. Below, we begin by describing the state of the art in OC in rigid body systems and MPC in legged robots.

A. Optimal control in rigid body systems

Forward dynamics fits naturally into classical optimal control formulations that include rigid body systems with constraints (e.g., holonomic contact constraints [16]), as we can condense the rigid body dynamics using Gauss’s principle of least constraint [17]. Moreover, this formulation can be solved efficiently via the DDP algorithm, as it exploits the temporal structure through Riccati factorization (also known as Riccati recursion) [18]. Both aspects increase computational efficiency because cache access is effective on smaller and dense matrices rather than large and sparse ones (see benchmarks in [19]). Indeed, state-of-the-art solvers for sparse linear algebra are not as efficient as factorizing through Riccati recursions (cf. [20]). Despite that, they are commonly used to solve optimal control problems with general-purpose numerical optimization programs such as SNOPT [21], KNITRO [22], and IPOPT [23].

In the case of optimal control with inverse dynamics, there is some evidence of the numerical benefits of such kinds of formulations. Concretely, there are two recent works based on direct transcription and inverse dynamics, which resolve the optimal control problem via a general-purpose nonlinear program [14] or a custom-made nonlinear optimal control solver [13]. The first work relies on KNITRO, an advanced general-purpose nonlinear solver, and benchmarks results with INTERIOR/DIRECT [24] and the sparse linear solvers provided by HSL [25] software: MA27, MA57, and MA97. The second work factorizes the linear system through Riccati recursion.
We name this approach \textsc{INTERIOR/RICCATI}. Both \textsc{INTERIOR/DIRECT} and \textsc{INTERIOR/RICCATI} techniques handle inequality constraints using primal-dual interior point and line search. However, \textsc{INTERIOR/RICCATI} is significantly faster (at least one order of magnitude) than \textsc{INTERIOR/DIRECT}. This is because the Riccati recursions exploit the problem’s temporal structure. Nevertheless, \textsc{INTERIOR/RICCATI} scales cubically to the number of equality constraints. This reduces its efficiency significantly for problems with contact constraints. Furthermore, as reported in [14], \textsc{INTERIOR/DIRECT} often struggles to get solutions with very low constraint satisfaction (i.e., lower than $10^{-3}$) in a few iterations. In this work, we provide a method that tackles these issues, i.e., an approach that converges fast and with high accuracy. Fig. 2 shows a sequence of optimized motions computed efficiently with our approach.

\section{MPC and legged locomotion}

Most state-of-the-art MPC approaches use reduced-order dynamics, such as the inverted pendulum model (e.g., [26], [27]), or single-rigid body dynamics (e.g., [28], [29], [30], [31]), which may also include the robot’s full kinematics [32], [33]. The motivation for doing this is to reduce computational complexity by ignoring the limb’s dynamics, as robots are often designed to have lightweight legs or arms. However, recent evidence suggests that the limb’s dynamics still play a significant role in the control of those types of robots (see [34]). In contrast, there is a wave of recent works that focus on MPC with the robot’s forward dynamics [3], [4], [5], [12]. Using the robot’s full-body dynamics brings benefits in terms of whole-body manipulation [3], [12] and agile locomotion and control [5], [35]. Nevertheless, to date, there is no single MPC approach that relies on inverse dynamics. This is due to the algorithmic complexity of handling equality constraints being much higher than its counterpart.

Feedback MPC computes local controllers that aim to increase tracking performance. These local controllers compensate for model errors and disturbances between MPC updates while considering the robot’s intrinsic properties. Building local controllers is challenging as simplifications of the robot’s dynamics tend to produce \textit{aggressive} controllers. For instance, when we assume that the robot behaves as a single-rigid body, the DDP algorithm computes feedback gains for a hypothetical system with higher bandwidth. This assumption produces feedback controllers that cannot stabilize the robot (cf. [33]). To deal with this issue, we can augment the dynamics with a filter, use full kinematics, and define a \textit{frequency-dependent} cost function as in [33]. However, this augmented system is larger than the full-body system. This in itself increases the time complexity of algorithms for optimal control. Alternatively, we can employ the robot’s rigid-body dynamics to model the bandwidth of the full-body system. Despite that, this model neglects the actuation bandwidth. This assumption seems to hold for a wide range of legged robots as we can build feedback controllers even for robots with serial elastic actuators such as ANYmal [5]. Note that the actuation bandwidth is lower in robots with elastic elements.

\section{Optimal control of rigid body systems}

In this section, we discuss the differences between forward and inverse optimal control formulations for rigid body systems subject to predefined contact sequences. We use a general problem formulation that considers arbitrary high-order integrators.

\subsection{Optimal control with the forward dynamics}

Classical optimal control formulations (e.g., [36]) involve the use of the forward dynamics:

\begin{equation}
\min_{(\hat{q}_k, \hat{v}_k, (\hat{\tau}, \hat{\lambda}))} \ell_N(q_N, v_N) + \sum_{k=0}^{N-1} \int_{t_k}^{t_{k+1}} \ell_k(q_k, v_k, \tau_k, \lambda_k) \, dt \\
\text{s.t. } [q_{k+1}, v_{k+1}] = \psi(q_k, v_k, FD(q_k, v_k, \tau_k, \lambda_k)), \\
q_k \in Q, \ v_k \in V, \ \tau_k \in \mathcal{P}, \ \lambda_k \in \mathcal{F},
\end{equation}

where $q_k$, $v_k$, $\tau_k$, and $\lambda_k$ are the decision variables and describe the configuration point, generalized velocity, joint effort commands, and contact forces of the rigid body system at node $k$; $\hat{q}_k$, $\hat{v}_k$, $\hat{\tau}_s$, and $\hat{\lambda}_s$ are vectors that stack the decision variables for all the nodes; $N$ defines the optimization horizon; $\ell(\cdot)$ describes the task as a cost function; $\psi(\cdot)$ defines the integrator function; $FD(\cdot)$ represents the forward dynamics, which computes the generalized accelerations $\hat{v}$ through the articulate body algorithm (ABA) [2]:

\begin{equation}
M \ddot{\mathbf{q}} = \tau_{bias} + J_c^T \lambda,
\end{equation}

or through the contact dynamics [16], [37]:

\begin{equation}
\begin{bmatrix}
\dot{\mathbf{q}} \\
-\lambda
\end{bmatrix} = \begin{bmatrix} M & J_c^T \\ J_c & 0 \end{bmatrix}^{-1} \begin{bmatrix} \tau_{bias} \\ -a_c \end{bmatrix}.
\end{equation}

Note that $M$ is the joint-space inertia matrix; $\tau_{bias}$ includes the torque inputs, Coriolis effect and gravity field; $a_c$ is the desired acceleration in the constraint space, which includes the Baumgarte stabilization [38]; $J_c$ is the stack of contact Jacobians (expressed in the local frame) that models the holonomic scleronomic constraints. Additionally, all the trajectories ($\hat{q}_k$, $\hat{v}_k$, $\hat{\tau}_s$, $\hat{\lambda}_s$) lie in their respective admissible sets. These admissible sets commonly define the robot’s joint limits, friction cone, task constraints, etc.

\subsection{Optimal control with the inverse dynamics}

In an inverse-dynamics OC formulation, we substitute the dynamical system with an equality constraint and include the generalized accelerations as decision variables:

\begin{equation}
[q_{k+1}, v_{k+1}] = \psi(q_k, v_k, a_k), \\
\text{ID}(q_k, v_k, a_k, \tau_k, \lambda_k) = 0,
\end{equation}

where $\text{ID}(\cdot)$ is the \textit{inverse-dynamics function}. This function is typically computed through the recursive Newton-Euler algorithm (RNEA) [2] if we do not consider the contact constraints $J_c \hat{v} - a_c = 0$. This formulation can be interpreted as a \textit{kinodynamic} motion optimization (e.g., [10]), in which we kinematically integrate the system while guaranteeing its dynamics through constraints. Note that the decision variables
for the $k$th node are now $q_k$, $v_k$, $a_k$, $\tau_k$, and $\lambda_k$, which increases compared to the forward-dynamics OC formulations.

A compelling motivation for optimal control approaches based on inverse dynamics is that they help solve coarse optimization problems, handle poor initialization, and improve the convergence rate. This is due to the numerical benefits of decoupling the integrator and robot dynamics. Indeed, as shown later in Section VII, our approach has the same advantages as reported in the literature.

C. Inverse vs forward dynamics in optimal control

We can compute the robot’s inverse dynamics more efficiently than the forward dynamics, as the algorithmic complexity of the RNEA is lower than the ABA [2]. The same applies to the computation of their derivatives [39], [9]. However, we cannot condense the decision variables if we are unwilling to sacrifice physical realism as in [40], [15]. This, unfortunately, increases the problem dimension and the computation time needed to factorize the Karush-Kuhn-Tucker (KKT) problem. In contrast, in the forward dynamics case, we can condense these variables thanks to the application of Gauss’s principle of least constraint [17]. Indeed, our recent work [37] showed highly-efficient computation of optimal control problems using inverse dynamics, which enables MPC applications in quadrupeds [5] and humanoids [12].

The increment in decision variables in an inverse-dynamics formulation is notable. For instance, it adds about 3000 extra decision variables (or 38%) for a quadruped robot and an optimization horizon of 100 nodes (or $1 \, s$). To solve these problems, we need to handle equality constraints efficiently and to further exploit the inverse-dynamics structure. However, the algorithmic complexity of state-of-the-art approaches scales cubically to the number of equality constraints.

D. Our nullspace approach in a nutshell

Fig. 3 shows a comparison between optimal control with forward dynamics and our inverse dynamics approach (blue blocks). Our method can be interpreted as a way to reduce system dimensionality, which solves faster optimal control problems with inverse dynamics. This reduction is due to (i) the nullspace parametrization in our OC solver and (ii) the condensed representation of inverse dynamics. Before introducing our condensed inverse dynamics, we begin by describing our nullspace method for solving efficiently OC problems with stagewise equality constraints. Our method uses exact algebra and exploits the problem’s structure as shown below.

IV. Resolution of optimal control with inverse dynamics

In this section, we describe our approach to handling inverse-dynamics (equality) constraints. These types of constraints are nonlinear and can be fulfilled within a single node i.e., stagewise. Here, we are interested in solving the following optimal control problem efficiently:

$$\min_{\hat{x}_N, \hat{u}_N} \ell_N(\hat{x}_N) + \sum_{k=0}^{N-1} \ell_k(\hat{x}_k, \hat{u}_k)$$

s.t. $x_0 = \bar{x}_0, \quad x_{k+1} = f_k(\hat{x}_k, \hat{u}_k), \quad h_k(\hat{x}_k, \hat{u}_k) = 0$, where, similarly to Eq. (4), $f(\cdot)$ defines the kinematic evolution of the system (i.e., integrator), $h(\cdot)$ ensures the feasibility of the dynamics through its inverse-dynamics function, $\bar{x}_0$ represents the initial condition of the system, $x := (q, v) \in \mathcal{X}$ is the state of the system and lies in a differential manifold (with dimension $n_x$), and $u \in \mathbb{R}^n_u$ is the input of the system that depends on the inverse-dynamics formulation as described in Section V. Note that $\hat{x}_k$ and $\hat{u}_k$ stack the state and control inputs of each node, respectively.

To describe our method, we begin by introducing the KKT conditions of Eq. (5) and the classical Newton method used to find its KKT point (Section IV-A). This boils down to a large saddle point problem that is often solved via sparse linear solvers. However, this is not the most efficient approach to solving this linear system of equations. Instead, we continue by developing a method that exploits the temporal structure inherent in optimal control (Section IV-B). Concretely, it breaks the large saddle point problem into a sequence of smaller sub-problems, as stated in Bellman’s principle of optimality and shown in Appendix A.

Our approach can be seen as a DDP algorithm [6] that handles stagewise equality constraints. To further improve efficiency, we introduce our nullspace parametrization and compare it with an alternative approach: the Schur complement (Section IV-C). This parametrization allows us to perform parallel computations, reducing the algorithmic complexity of solving the saddle point problem. Finally, we explain how our method drives the dynamics and equality constraint...
infeasibilities to zero (Section IV-E). It boils down to a novel combination of feasibility-driven search and a merit function.

A. Optimality conditions

To obtain the local minimum of Eq. (5), we need to find the KKT point, i.e., a point where the KKT conditions hold. Essentially, these are first-order necessary conditions for determining the stationary point of the problem’s Lagrangian and ensuring the primal feasibility, i.e.,

\[
\nabla_{\hat{w}} \mathcal{L}(\hat{w}_a, \hat{\gamma}_a, \hat{\xi}_a) = 0, \quad \text{(stationary condition)} \quad (6)
\]

\[
\check{x}_k(x_k, u_k) = 0, \quad \forall k = \{0, \ldots, N - 1\} \quad (7)
\]

where \(\hat{w}_a := (\hat{x}_a, \hat{u}_a)\) is the stack of primal variables (i.e., state and control of each node), \((\hat{\gamma}_a, \hat{\xi}_a)\) is the stack of Lagrange multipliers associated with the initial condition, system dynamics and equality constraint, \(\odot\) is the difference operator needed to optimize over manifolds [41], and the Lagrangian of Eq. (6) is defined as:

\[
\mathcal{L}(\hat{w}_a, \hat{\gamma}_a, \hat{\xi}_a) = \hat{c}_a(\hat{w}_a) + \hat{\gamma}_a^T \hat{h}_a(\hat{w}_a) + \hat{\xi}_a^T \hat{d}_a(\hat{w}_a), \quad (8)
\]

with the above terms as

\[
\hat{c}_a(\hat{w}_a) = \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k),
\]

\[
\hat{\gamma}_a^T \hat{h}_a(\hat{w}_a) = \sum_{k=0}^{N-1} \hat{\gamma}_a^T \hat{h}_k(x_k, u_k),
\]

\[
\hat{\xi}_a^T \hat{d}_a(\hat{w}_a) = \xi_0^T (\hat{x}_0 \odot x_0) + \sum_{k=0}^{N-1} \xi_a^T (f_k(x_k, u_k) \odot x_{k+1}).
\]

Note that the notation introduced to optimize over manifolds is inspired by [42] and adopted in CROCODY [37].

To compute a triplet \((\hat{w}_a, \hat{\gamma}_a, \hat{\xi}_a)\) satisfying the roots of Eq. (6) and Eq. (7), we apply the Newton method leading to:

\[
\begin{bmatrix}
\nabla^2_{\hat{w}_a} \mathcal{L} & \nabla \hat{h}_a & \nabla \hat{d}_a \\
\nabla \hat{h}_a & \nabla^2 \hat{\gamma}_a & \nabla \hat{\xi}_a \\
\nabla \hat{d}_a & \nabla \hat{\xi}_a & \nabla^2 \hat{x}_a \\
\end{bmatrix}
\begin{bmatrix}
\delta \hat{w}_a \\
\delta \hat{\gamma}_a \\
\delta \hat{\xi}_a \\
\end{bmatrix}
\begin{bmatrix}
\nabla \hat{c}_a \\
\nabla \hat{h}_a \\
\nabla \hat{d}_a \\
\end{bmatrix}
= - \begin{bmatrix}
\nabla \check{c}_a \\
\nabla \check{h}_a \\
\nabla \check{d}_a \\
\end{bmatrix},
\quad (9)
\]

where \(\delta \hat{w}_a\) is the search direction computed for the primal variables, \(\hat{\gamma}_a^+ := \hat{\gamma}_a + \delta \hat{\gamma}_a\) and \(\hat{\xi}_a^+ := \hat{\xi}_a + \delta \hat{\xi}_a\) are the next values of the Lagrange multipliers. Despite the use of sparse linear solvers, this generic method does not exploit the temporal structure of the optimal-control problem. Below, we describe how our approach exploits the temporal structure, leading to an efficient factorization of the saddle point problem in Eq. (9).

B. Equality constrained differential dynamic programming

If we apply the Bellman principle of optimality to the saddle point problem defined in Eq. (9), we can now solve recursively the following system of linear equations per each node \(k\):

\[
\begin{bmatrix}
\mathcal{L}_{xx} & \mathcal{L}_{ux} & \hat{h}_x^T & \hat{f}_x^T \\
\mathcal{L}_{ux} & \mathcal{L}_{uu} & \hat{h}_u^T & \hat{f}_u^T \\
\hat{h}_x & \hat{h}_u & -I & \gamma^+ \\
\hat{f}_x & \hat{f}_u & -I & \gamma^+ \\
\end{bmatrix}
\begin{bmatrix}
\delta \hat{x}_k \\
\delta \hat{u}_k \\
\gamma_k^+ \\
\gamma_k^+ \\
\end{bmatrix}
= - \begin{bmatrix}
\ell_x \\
\ell_u \\
\hat{h} \\
\hat{f} \\
\end{bmatrix}, \quad (10)
\]

with

\[
\mathcal{L}_{xx} := \ell_{xx} + \gamma_x^+ \mathcal{L}_{xx}, \quad \mathcal{L}_{uu} := \ell_{uu} + \gamma_u^+ \mathcal{L}_{uu}, \quad \gamma^+ := \gamma + \delta \gamma,
\]

\[
\mathcal{L}_{xx} \in \ell_{xx} + \gamma_x^+ \mathcal{L}_{xx}, \quad \mathcal{L}_{uu} \in \ell_{uu} + \gamma_u^+ \mathcal{L}_{uu}, \quad \gamma^+ := \gamma + \delta \gamma,
\]

where \(\ell_p, h_p, f_p\) correspond to the first derivatives of the cost, equality constraint, and system dynamics with respect to \(p\), respectively; \(f_{pp}\) is the second derivative of the system dynamics; \(\gamma_x^+, \gamma_u^+\) are the gradient and Hessian of the value function; \(\gamma, \xi\) are the Lagrange multipliers associated with the equality constraint and system dynamics; \(\hat{h}, \hat{f}\) describe the gaps in the equality constraint and dynamics; \(\delta \hat{x}, \delta \hat{u}, \delta x^+\) and \(\delta \gamma, \delta \xi\) provides the search direction computed for the primal and dual variables, respectively; \(\gamma_x^+ \cdot f_{pp}\) defines the tensor product with the Hessian of the system dynamics. Finally, note that (i) \(p\) is a hypothetical decision variable that represents \(x\) or \(u\), (ii) we have dropped the node index \(k\) and introduced \('\) notation to describe the node index \(k+1\), and (iii) \(\tilde{f}\) corresponds to the kinematic gap while \(\hat{h}\) refers to the inverse-dynamics and contact-acceleration gaps.

1) Where does this equation come from?: Eq. (10) exploits the temporal structure of the optimal control problem, as it breaks this large problem into smaller sub-problems. Those sub-problems are solved recursively and backwards in time. We can do so because the following relationship holds for all the nodes

\[
\mathcal{L}_{xx} := \ell_{xx} + \gamma_x^+ \mathcal{L}_{xx}, \quad \mathcal{L}_{uu} := \ell_{uu} + \gamma_u^+ \mathcal{L}_{uu}, \quad \gamma^+ := \gamma + \delta \gamma,
\]

as shown in Appendix A. This equation connects the derivatives of the value function with the next costate \(\xi^+\). Such a connection should not surprise us if we observe that Pontryagin’s maximum principle (PMP) and KKT conditions are two equivalent ways to define the local minima. Indeed, Bellman recognized this connection in his groundbreaking work [43], which is better known for establishing the Hamilton-Jacobi-Bellman (HJB) equation in the continuous-time domain. Alternatively, we encourage the readers to see [18] that revisits this connection. From now on, we will elaborate on the different concepts used by our algorithm.

C. Search direction and factorization

We start by condensing the fourth and fifth rows of Eq. (10), which yields to

\[
\begin{bmatrix}
Q_{xx} & Q_{ux} & h_x^T & f_x^T \\
Q_{ux} & Q_{uu} & h_u^T & f_u^T \\
h_x & h_u & -I & \gamma^+ \\
f_x & f_u & -I & \gamma^+ \\
\end{bmatrix}
\begin{bmatrix}
\delta \hat{x} \\
\delta \hat{u} \\
\gamma^+ \\
\gamma^+ \\
\end{bmatrix}
= - \begin{bmatrix}
Q_x \\
Q_u \\
\hat{h} \\
\hat{f} \\
\end{bmatrix}, \quad (12)
\]
where the $Q$'s describe the local approximation of the action-value function in the free space:

$$
\begin{align*}
Q_x &= \ell_x + f_x^T \nu_x^+, \\
Q_{xx} &= \ell_{xx} + f_{xx}^T \nu_{xx}, \\
Q_{ux} &= \ell_{ux} + f_{ux}^T \nu_{xx}, \\
Q_{uu} &= \ell_{uu} + f_{uu}^T \nu_{xx},
\end{align*}
$$

with $\nu_x^+ := \nu_x + \nu_{xx} \bar{f}$ representing the gradient of the value function after the deflection produced by the dynamics gap $\bar{f}$ (see [37], [18]). Furthermore, we apply the DDP approach which means we express changes in the control inputs $\delta u$ as a function of changes in the state of the system $\delta x$. This choice can be interpreted as minimizing Eq. (12) with respect to $\delta u$ only, i.e.,

$$
\Delta v = \min_{\delta u} \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{ux} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} + \begin{bmatrix} \delta x \end{bmatrix}^T \begin{bmatrix} Q_x \end{bmatrix},
$$

s.t. $[h_x \ h_u] \begin{bmatrix} \delta x \\ \delta u \end{bmatrix} + \bar{h} = 0,$

(14)

where this local quadratic program has the following first-order necessary conditions of optimality

$$
\begin{bmatrix} Q_{uu} & h_u^T \\ h_u & \bar{h} \end{bmatrix} \begin{bmatrix} \delta u \\ \gamma^+ \end{bmatrix} = - \begin{bmatrix} Q_u + Q_{ux} \delta x \end{bmatrix},
$$

(15)

Note that this equation forms a dense and small saddle point system, which we solve for each node.

Eq. (15) can be solved by what we call a Schur-complement factorization (e.g., [44]). However, this approach increases the algorithmic complexity of the Riccati recursion. This increment is related to the number of equality constraints. Instead, we propose a nullspace factorization approach that does not increase the algorithmic complexity needed to handle equality constraints. This is particularly relevant in inverse-dynamics formulations, as these problems have a large number of equality constraints. Below, we first introduce the Schur-complement factorization and then our nullspace factorization. With this, we explain the drawbacks of the Schur-complement approach and then justify the computational benefits of our method.

1) Schur-complement factorization: We can compute the control policy (i.e., the search direction for primal decision variable $\delta u$) by factorizing Eq. (15) through the Schur-complement approach:

$$
\begin{align*}
\delta u &= -k - K \delta x - \Psi_{\gamma}, \\
\gamma^+ &= \tilde{Q}_{uu}(k + K \delta x),
\end{align*}
$$

(16)

where $k = Q_{uu}^{-1} Q_{u}$, $K = Q_{uu}^{-1} Q_{ux}$ are the feed-forward and feedback gain on the free space, $\Psi_{\gamma} = Q_{uu}^{-1} h_u$, $\tilde{Q}_{uu} = (h_u Q_{uu}^{-1} h_u) ^{-1}$, $k_s = \bar{h} - h_u k$, $K_s = h_x - h_u K$ are terms associated with the computation of the feed-forward and feedback gain on the constrained space. Thus, the final control policy has the form

$$
\delta u = -\pi - \Pi \delta x,
$$

with

$$
\begin{align*}
\pi &= k + (k_s^T \tilde{Q}_{uu} \Psi_{\gamma}^T)^T \quad \text{(feed-forward)}, \\
\Pi &= K + (K_s^T \tilde{Q}_{uu} \Psi_{\gamma}^T)^T \quad \text{(feedback gain)}.
\end{align*}
$$

(17)

This factorization technique requires performing two Cholesky decompositions for computing $Q_{uu}^{-1}$ and $Q_{uu}$. It means that the computational complexity of obtaining the control policy increases cubically with respect to the number of equality constraints as well. Note that $Q_{uu}$ is a square matrix with a dimension equal to the number of equality constraints of the node.

2) Nullspace factorization: Using the fundamental basis of $h_u$, we can decompose/parametrize the decision variable $\delta u$ as follows

$$
\delta u = Y \delta u_y + Z \delta u_z,
$$

(18)

where $Z \in \mathbb{R}^{n_x \times n_z}$ is the nullspace basis of $h_u$ (with $n_z$ as its nullity), $Y$ is chosen such that $[Y Z]$ spans $\mathbb{R}^{n_u}$. Then, by substituting $\delta u$ into the above optimality conditions, observing that $h_u Z = 0$ and premultiplying by $Z^T$, we obtain

$$
\begin{align*}
Z^T Q_{uu} Z \quad Z^T Q_{uu} Y \quad \delta u_y \quad \delta u_y \\
\delta u_y \quad \delta u_y \quad - Z^T (Q_{uu} + Q_{ux} \delta x) \quad h + h_x \delta x
\end{align*},
$$

(19)

which is a reduced saddle point system, as it removes the need to compute $\gamma^+$ in Eq. (14). Indeed, Eq. (19) allows us to compute the control policy as:

$$
\delta u = -\pi - \Pi \delta x,
$$

with

$$
\begin{align*}
\pi &= zk + \tilde{Q}_{zz} \Psi_{\gamma} \bar{h} \quad \text{(feed-forward)}, \\
\Pi &= zK + \tilde{Q}_{zz} \Psi_{\gamma} h_x \quad \text{(feedback gain)},
\end{align*}
$$

(20)

where $k_z = Q_{zz}^{-1} Q_z$, $K_z = Q_{zz}^{-1} Q_{zz}$ are the feed-forward and feedback gain associated with the nullspace of the equality constraint, $\tilde{Q}_{zz} = I - Z Q_{zz}^{-1} Z_z$ are terms that project the constraint into both spaces: range and nullspace. Note that $Q_z = Z^T Q_u$, $Q_{zz} = Z^T Q_{uu}$, $Q_{uu} = Z^T Q_{uu}$ describe the local approximation of the action-value function in the nullspace.

Now, we observe that our nullspace factorization requires performing three decompositions for computing $Q_{zz}^{-1}$, $(h_u Y)^{-1}$, and the constraint basis for the image and nullspace $[Y Z]$. The two formers can be inverted efficiently via the Cholesky and lower-upper (LU) with partial pivoting decompositions, respectively, as these matrices are positive definite and square invertible by construction. Instead, the constraint basis can be computed using any rank-revealing decomposition such as LU with full pivoting or QR with column pivoting [45].

3) Why is our nullspace approach more efficient?: Although the nullspace factorization requires an extra decomposition compared with the Schur-complement approach (i.e., it increases the algorithm complexity), its computation can be parallelized leading to a reduced computational cost. Concretely, $\Psi_{\gamma} = Y(h_u Y)^{-1}$, $\Psi_{\gamma} \bar{h}$, $\Psi_{\gamma} h_x$, and $[Y Z]$ are computed in parallel as their computations do not depend on the derivatives of the value function. Thus, the computational complexity of the Riccati recursion does not grow with the number of equality constraints, as it runs in the nullspace of the constraints. Indeed, its asymptotic complexity is $O(N(n_x + n_z)^3)$, where $n_z$ defines the dimension of the state and $n_u$ of the control vector. Furthermore, these computations can be...
sequential
where
π
on the Jacobians of the equality constraint as well.

in [47]) because the feed-forward and feedback terms depend
this unconstrained case, we cannot simplify its expression (as
is similar to the unconstrained DDP. However, in contrast to
D. Value function in a single node

As explained in Appendix A, the quadratic approximation
of value function in a given node is

\[ \Delta V = \Delta V_1 + \frac{\Delta V_2}{2} + \delta x^T V_x + \frac{1}{2} \delta x^T V_{xx} \delta x \]  \hspace{1cm} (21)

with

\[ \Delta V_1 = -\pi^T Q_u, \quad \Delta V_2 = \pi^T Q_{uu} \pi, \]
\[ V_x = Q_x + \Pi^T (Q_{uu} \pi - Q_u) - Q_{ux} \pi, \]
\[ V_{xx} = Q_{xx} + (\Pi^T Q_{uu} - 2 Q_{ux} \Pi) \Pi, \]  \hspace{1cm} (22)

where \( \pi, \Pi \) can be computed with Eq. (20) or (17). Note that,
in essence, computing this approximation of the value function
is similar to the unconstrained DDP. However, in contrast to
this unconstrained case, we cannot simplify its expression (as
in [47]) because the feed-forward and feedback terms depend
on the Jacobians of the equality constraint as well.

E. Nonlinear step and merit function

Instead of a classical line-search procedure (see [48, Chapter
3]), we try the search direction along a feasibility-driven
nonlinear rollout of the dynamics:

\[ \begin{align*}
x^{t+1}_k & = x^t_k + \alpha \delta x^t_k, \\
u^{t+1}_k & = \nu^t_k + \alpha \delta \nu^t_k, \end{align*} \]  \hspace{1cm} (23)

for \( k = \{0, \ldots, N-1 \} \) and \( \delta x^t_k = x^{t+1}_k - x^t_k \), where a backtracking procedure tries different step lengths \( \alpha \); \( x^t_k, \nu^t_k \)
describes the potential new guess for the \( k \)th node; \( \bar{f}_0 \) is the
gap of the initial state condition; \( \oplus \) is the integrator operator
needed to optimize over manifolds [41]. Our feasibility-driven
nonlinear rollout reduces dynamic infeasibility as expected in
direct multiple shooting formulations, i.e., based on the step
length (see [18, Section II]). As described below, we compute a
feasibility-aware expected improvement, which allows us to
evaluate a given step length.

To evaluate the goodness of a given step length \( \alpha \in (0,1] \),
we first compute the expected cost reduction using the local
approximation of the value function. However, the quadratic
approximation in Eq. (22) does not consider the expected evo-
lation of the gaps in the system dynamics \( \bar{f}_k = (\bar{f}_0, \ldots, \bar{f}_N) \),
which is critical for increasing the basin of attraction to
good local minima of our algorithm and for generating agile
acrobatic maneuvers (see Section VII and [18]). Therefore, to
account for the multiple-shooting effect of \( \bar{f}_k \), we compute a
feasibility-aware expected improvement as

\[ \Delta \ell(\alpha) = \alpha \sum_{k=0}^{N-1} \left( \Delta \ell_{1k} + \frac{1}{2} \alpha \Delta \ell_{2k} \right), \]  \hspace{1cm} (24)

where, by closing the gaps as proposed in [37] in the linear
rollout, we have:

\[ \begin{align*}
\Delta \ell_{1k} & = \pi_k^T Q_{uu} \nu_{uk} + \bar{f}_k^T (V_{xk} - V_{xuk} \delta x^t_k), \\
\Delta \ell_{2k} & = \pi_k^T Q_{uuk} \pi_k + \bar{f}_k^T (2V_{xuk} \delta x^t_k - V_{xuk} \delta u_k). \end{align*} \]  \hspace{1cm} (25)

This expected improvement matches the quadratic approxima-
tion of the value function in Eq. (22) if there is no dynamics
infeasibility, i.e., \( \bar{f}_k = 0 \) for all \( 0 < k < N \).

We then compute a merit function of the form:

\[ \phi(\bar{x}_k, \bar{u}_k; \nu) = \sum_{k=0}^{N-1} \ell(x_k, u_k) + \nu \epsilon(x_k, u_k), \]  \hspace{1cm} (26)

with

\[ \epsilon(x_k, u_k) = \| \bar{f}_k \|_1 + \| \bar{h}_k \|_1, \]

where \( \epsilon(\cdot) \) measures the infeasibility of the current guess at
the \( k \)th node, and \( \nu \) is the penalty parameter that balances
optimality and feasibility. We update this parameter at every
iteration as follows

\[ \nu \leftarrow \max \left( \nu, \frac{\Delta \ell(1)}{(1 - \rho) \sum_{k=0}^{N-1} \epsilon(x_k, u_k)} \right), \]  \hspace{1cm} (27)
given \( 0 < \rho < 1 \), which is a tunable hyper-parameter.
Our updating rule is inspired by [49], where \( \Delta \ell(1) \) can be
interpreted as the objective function of the tangential subproblem and \( \sum_{k=0}^{N-1} \epsilon(x_k, u_k) \) as the reduction provided by the normal step.

Finally, we accept a step \( \delta \hat{w}_s^\alpha = \alpha [\delta x_s^T \delta u_s^T]^T \) if the following Goldstein-inspired condition holds:

\[
\phi(\hat{w}_s^+; \nu) - \phi(\hat{w}_s^-; \nu) \leq \eta_1 \Delta \Phi(\hat{w}_s; \nu) \quad \text{if} \quad \Delta \Phi(\hat{w}_s; \nu) \leq 0
\]

\[
\eta_2 \Delta \tilde{\ell}(\alpha) \quad \text{otherwise}
\]

with \( \Delta \Phi(\hat{w}_s; \nu) = D(\phi(\hat{w}_s; \nu); \delta \hat{w}_s^\alpha) \), where, again, \( \hat{w}_s \) contains the current (guess) state and control trajectories \( \hat{x}_s, \hat{u}_s \), \( \hat{w}_s^+ := \hat{w}_s + \alpha \delta \hat{w}_s \) is the next guess under trial, \( 0 < \eta_1 < 1 \) and \( 0 < \eta_2 \) are user-defined parameters, and

\[
D(\phi(\hat{w}_s; \nu); \delta \hat{w}_s^\alpha) := \Delta \ell(\alpha) + \alpha \sum_{k=0}^{N-1} \epsilon(x_k, u_k)
\]

denotes the directional derivative of \( \phi(\cdot) \) along the direction \( \delta \hat{w}_s^\alpha \). Our Goldstein-inspired condition allows the algorithm to accept ascend directions in \( \Delta \Phi(\hat{w}_s; \nu) \), which might occur during iterations that are dynamically infeasible (i.e., \( \bar{F}_s \neq 0 \)). Note that we use \( \eta_2 \Delta \tilde{\ell}(\alpha) \) in the second condition, as it quantifies the dynamics infeasibility only.

F. Regularization and stopping criteria

To increase the algorithm’s robustness, we regularize \( V_{xx}^\nu \) and \( L_{uu} \) in such a way that changes the search direction from Newton to steepest descent conveniently and handles potential concavity. Concretely, our regularization procedure follows a Levenberg-Marquardt scheme [50] to update \( \mu \), i.e.,

\[
\mu \leftarrow \beta^d \mu, \quad V_{xx}^\nu \leftarrow V_{xx}^\nu + \mu \bar{f}_x^T \bar{f}_x, \quad L_{uu} \leftarrow L_{uu} + \mu I
\]

where \( \beta^d \) and \( \beta^u \) are factors used to increase and decrease the regularization value \( \mu \), respectively. Our updating rule is as follows: we increase \( \mu \) when the Cholesky decomposition in Eq. (20) (or in Eq. (17)) fails or when the forward pass accepts a step length smaller than \( \alpha_0 \) or \( \sum_{k=0}^{N-1} \ell_{2,k} \) is lower than \( \kappa_0 \leq 10^{-4} \). This latter condition allows the algorithm to focus on reducing the infeasibility in the equality constraints after reaching a certain level of optimality. Instead, we decrease \( \mu \) when the forward pass accepts steps larger than \( \alpha_1 \).

The \( V_{xx}^\nu \) term is defined as \( L_{xx} + \mu \bar{f}_x^T \bar{f}_x \), which can be also interpreted as a banded regularization as it corrects the matrix inertia that condenses the future nodes (i.e., \( V_{xx}^\nu \)). These type of strategies for inertia correction are similarly implemented in general-purpose nonlinear programming solvers such as IPOPT [23].

1) Stopping criteria: We consider both optimality and feasibility in the stopping criteria by employing the following function:

\[
\max \left( \sum_{k=0}^{N-1} \epsilon(x_k, u_k), |\Delta \ell(1)| \right).
\]

We say that the algorithm has converged to a local minimum when the value of the above stopping criterion function is lower than a user-defined tolerance, which is \( 10^{-9} \) in our results.

Algorithm 1: Equality constrained DDP

```plaintext
/* derivatives and decompositions */
for k ← 0 to N do in parallel
    residual and derivatives: \( \bar{f}_x, \bar{f}_p, \ell_p, L_{pp}, f_p, h_p \)
if nullspace factorization then
    nullspace and decomposition: \( Y, Z, (h_u Y)^{-1} \)
    feed-forward/back in rank-space: \( \Psi_n \bar{h}, \Psi_n h_x \)
/* compute search direction */
for k ← N - 1 to 0 do
    local action-value function
    regularization: \( V_{xx}^\nu, L_{uu} \)
if nullspace factorization then
    nullspace action-value: \( Q_{xx}, Q_{zz}, Q_{zx}, Q_{zu} \)
    nullspace Cholesky: \( Q_{xx}^{-1} \)
    feed-forward and feedback: \( \pi_n, \Pi_n \)
else
    full-space Cholesky: \( Q_{xx}^{-1}, k, K \)
    projected Cholesky: \( (h_u Q_{xx}^{-1} h_u)^{-1} \)
    free-space terms: \( \Psi_n, k_s, K_s \)
    feed-forward and feedback: \( \pi_s, \Pi_s \)
value function: \( \Delta V, V_{xx}, V_{zx} \)
/* try search direction */
for \( \alpha \in \{1, \frac{1}{2}, \ldots, \frac{1}{m} \} \) do
    nonlinear rollout: \( x_k^+, u_k^+ \)
    expected improvement: \( |\Delta \ell(\alpha)| \)
    merit penalty parameter: \( \nu \)
    merit function: \( \phi(x_s, u_s; \nu) \)
if success step then
    go to 1 until convergence
```

G. Algorithm summary

Algorithm 1 summarizes our novel equality-constrained DDP algorithm. It considers infeasibility for both dynamics and equality constraints and includes the Schur-complement and nullspace factorizations. As described above, the complexity of the Schur-complement factorization scales with respect to the dimension of the equality constraints. This is obvious if we observe that it requires performing a “projected” Cholesky decomposition (line 16) as well. Instead, our nullspace factorization performs a single Cholesky decomposition with the dimension of the kernel of \( h_u \), which is lower than the \( n_u \) (i.e., the dimension of the full-space Cholesky in line 15).

Our nullspace parametrization provides a competitive alternative compared to barrier methods such as augmented Lagrangian [51, 52, 53] or interior-point [13, 54] as well. The reasons are due to (i) the reduction in the dimension of the matrix needed to be decomposed using the Cholesky method (see again Fig. 4), and (ii) the use of exact algebra to handle these types of constraints. This factorization also allows us to perform partial computation in parallel of \( Y, Z, (h_u Y)^{-1}, \Psi_n \bar{h}, \) and \( \Psi_n h_x \) (lines 4 and 5) terms. We try the search
direction with a nonlinear step (based on system rollout), as it leads to faster convergence in practice. Similar empirical results have been reported in [55] for unconstrained problems. Finally, in each iteration, we update the penalty parameter in the merit function to balance optimality and feasibility (lines 24 and 25).

V. FUNCTIONAL STRUCTURE OF INVERSE DYNAMICS PROBLEM

As described in Section III-B, the decision variables are $\mathbf{q}$, $\mathbf{v}$, $\mathbf{a}$, $\mathbf{\tau}$, and $\mathbf{\lambda}$. The first couple of variables naturally describe the state of the system $\mathbf{x} = [\mathbf{q}^\top \mathbf{v}]^\top$, while the latter ones refer to its control inputs $\mathbf{u} = [\mathbf{a}^\top \mathbf{\tau}^\top \mathbf{\lambda}]^\top$. Therefore, the linearization of the nonlinear equality constraints has the form:

$$
\begin{bmatrix}
\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}} & \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}} \\
-\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{q}} & -\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{v}}
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{q} \\
\delta \mathbf{v}
\end{bmatrix}
+ \\
\begin{bmatrix}
\mathbf{h}_{\mathbf{u}^{\text{collected}}} \\
\mathbf{h}_{\mathbf{u}^{\text{collected}}}
\end{bmatrix}
= - \begin{bmatrix}
\mathbf{\tilde{h}}_{\mathbf{ID}} \\
\mathbf{\tilde{h}}_{\mathbf{A}}
\end{bmatrix}
$$

(31)

with

$$
\begin{aligned}
\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}} &= \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}}, \\
\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}} &= \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}}, \\
\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{q}} &= \frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{q}}, \\
\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{v}} &= \frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{v}},
\end{aligned}
$$

where $\mathbf{\tilde{h}}_{\mathbf{ID}}$, $\mathbf{\tilde{h}}_{\mathbf{A}}$ are the values of joint efforts and contact acceleration at the linearization point; $\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}}$, $\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}}$ are the RNEA derivatives [9]; $\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{q}}$, $\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{v}}$ are the derivatives of the frame acceleration; and $\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{q}}$, $\frac{\partial \mathbf{h}_\mathbf{\tau}}{\partial \mathbf{v}}$ are the derivatives of the arbitrary actuation model $\mathbf{A}(\mathbf{q}, \mathbf{v}, \mathbf{\tau})$. The dimension of the control input is $n_e + n_j + n_f$ with $n_e$ as the dimension of the generalized velocity, $n_j$ as the number of joints, and $n_f$ as the dimension of the contact-force vector. However, it is possible to condense this equation. This in turn reduces computation time as the algorithmic complexity depends on the dimension of $\mathbf{u}$. We call this new formulation condensed, while the above one is redundant. Below we will provide more details about the condensed inverse dynamics.

A. Condensed inverse dynamics

Including both joint efforts and contact forces creates redundancy and increases sparsity. However, it is more efficient to condense them as the asymptotic complexity of our equality-constrained DDP algorithm depends on the dimension of the system’s dynamics (i.e. $n_e + n_u$). We can do so by assuming that the RNEA constraint is always feasible (i.e., $\mathbf{\tilde{h}}_{\mathbf{ID}} = 0$), which leads to the following expression

$$
\delta \mathbf{\tau} = \left(\frac{\partial \mathbf{A}}{\partial \mathbf{\tau}}\right)^\top \left(\frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}} \delta \mathbf{q} + \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}} \delta \mathbf{v}\right),
$$

(32)

and if we plug this expression in Eq. (31), then we obtain a condensed formulation:

$$
\begin{bmatrix}
\mathbf{S} \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{q}} \\
\mathbf{S} \frac{\partial \mathbf{h}_\mathbf{a}}{\partial \mathbf{v}}
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{q} \\
\delta \mathbf{v}
\end{bmatrix}
+ \\
\begin{bmatrix}
\mathbf{M} - \mathbf{J}_c^\top \mathbf{h}_{\mathbf{u}^{\text{collected}}} \\
\mathbf{J}_c^\top
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{a} \\
\delta \mathbf{\tau}
\end{bmatrix}
= - \begin{bmatrix}
\mathbf{\tilde{h}}_{\mathbf{ID}} \\
\mathbf{\tilde{h}}_{\mathbf{A}}
\end{bmatrix}
$$

(33)

with $\mathbf{\tilde{h}}_{\mathbf{A}}$ as the value of the under-actuated efforts at the linearization point and $\mathbf{S} = \mathbf{I} - \frac{\partial \mathbf{A}}{\partial \mathbf{\tau}} \left(\frac{\partial \mathbf{A}}{\partial \mathbf{\tau}}\right)^\top$ as the matrix that selects under-actuated joints. This matrix is often constant in robotic systems (e.g., quadrotors, manipulators, and legged robots) and can be computed at once. Moreover, $\frac{\partial \mathbf{A}}{\partial \mathbf{\tau}}$ is not always a square matrix as in the case of quadrotors, but still, we can compute the selection matrix $\mathbf{S}$ via the pseudoinverse.

We now eliminate the joint efforts and reduce the dimension of the control input to $\mathbf{u} = [\mathbf{a}^\top \mathbf{\tau}^\top \mathbf{\lambda}]^\top$. Although a similar procedure was described in [13], this approach does not consider arbitrary actuation models such as propellers in quadrotors.

B. Exploiting the functional structure

The control inputs for both inverse-dynamics formulations define a functional sparsity, which does not appear in optimal control for forward-dynamics cases. Concretely, this sparsity is in the partial derivatives of system dynamics (i.e., integrator) with respect to $\mathbf{u}$, i.e.,

$$
\mathbf{f}_\mathbf{u} = \begin{bmatrix} \mathbf{f}_a \\ 0 \end{bmatrix},
$$

(34)

where $\mathbf{f}_a$ is the partial derivatives with respect to the generalized accelerations, and the null block represents the partial derivatives of the remaining components (i.e., $\mathbf{\tau}$ and/or $\mathbf{\lambda}$). Furthermore, the structure of $\mathbf{f}_a$ depends on the configuration manifold and chosen integrator. We inject this structure into the computation of the local approximation of the action-value function. Thus, we adapt the terms in Eq. (13) as follows

$$
\begin{aligned}
\mathbf{f}_\mathbf{u}^\top \nabla_x^+ &= \begin{bmatrix} \mathbf{f}_a^\top \nabla_x^+ \\ 0 \end{bmatrix}, \\
\mathbf{f}_\mathbf{u}^\top \nabla_{xx} \mathbf{f}_\mathbf{x} &= \begin{bmatrix} \mathbf{f}_a^\top \nabla_{xx} \mathbf{f}_\mathbf{x} \\ 0 \end{bmatrix}, \\
\mathbf{f}_\mathbf{u}^\top \nabla_{xx} \mathbf{f}_\mathbf{u} &= \begin{bmatrix} \mathbf{f}_a^\top \nabla_{xx} \mathbf{f}_\mathbf{u} \\ 0 \end{bmatrix}.
\end{aligned}
$$

(35)

From now on, we will describe our inverse-dynamics MPC, which is the heart of our perceptive locomotion pipeline. This MPC formulation is possible thanks to the computational advantages of our equality-constrained DDP algorithm.

VI. INVERSE-DYNAMICS MPC AND PIPELINE

Our inverse-dynamics MPC is formulated via time-based hybrid dynamics as described above. This formulation is inspired by our previous work [5]. But, before describing our MPC approach, we introduce our perceptive locomotion pipeline.
A. Perceptive locomotion pipeline

The core of our perceptive locomotion pipeline is a unique inverse-dynamics MPC approach that computes control policies, maps them to the joint-effort space, and sends them to a feedback-policy controller. Our inverse-dynamics MPC is designed to track the footstep plans given a reference velocity command from a joystick and a terrain map perceived by a depth camera. This allows us to avoid the ill-posed nature of the nonlinear complementary constraints (e.g. [56], [57], [58], [59]). Fig. 5 illustrates the different modules of our locomotion pipeline used to evaluate our inverse-dynamics MPC. Note that we briefly describe each module below for the sake of clarity. However, more details are presented in [7].

1) Terrain elevation and segmentation: We represent the terrain through a set of safe footstep regions similar to [60], [61], [62], [63]. These regions are computed from a terrain elevation map, which is commonly used in locomotion frameworks, e.g., [64], [65], [66], [67]. Concretely, using a depth camera, we build online a local terrain elevation map around the robot [68]. To do so, we fuse proprioceptive information (IMU and leg odometry) with LiDAR localization at 400 Hz. Finally, given the elevation map and the robot pose in the odometry frame, we extract a set of convex surfaces suitable for selecting the footstep regions.

2) Footstep placement and region selection: We break the combinatorics associated with the discrete choice of footstep regions when selecting footstep placements. Our approach first selects footstep regions and then determines footstep regions when selecting footstep placements. Our approach combinatorics associated with the discrete choice of footstep placements.

3) Inverse-dynamics MPC and feedback policy: The inverse-dynamics MPC (our contribution) computes whole-body motions and contact forces given a predefined footstep plan as follows:

\[
\min_{\hat{x}, \hat{u}} \sum_{k=0}^{N-1} \|q_k \ominus q_{\text{nom}}\|_Q^2 + \|v_k\|_N^2 + \|u_k\|_R^2 + \|\lambda_{c_k}\|^2_K
\]

s.t. \(q_0 = \hat{q}_0\), \(v_0 = \hat{v}_0\), (initial pos.)

if contact-gain transition:

\[
q_{k+1} = q_k, \\
\left[ \begin{array}{c} v_{k+1} \\ -\lambda_{c_k} \end{array} \right] = \left[ \begin{array}{c} M_k \\ -J^\top C_k \end{array} \right]^{-1} \left[ \begin{array}{c} \tau_{\text{bias},k} \\ -\alpha_{c_k} \end{array} \right],
\]

else:

\[
\tau_{k+1}, v_{k+1} = \mathbf{\psi}(q_k, v_k, a_k),
\]

\(I^D(q_k, v_k, a_k, \lambda_{c_k}) = 0\),

\(\lambda_{\nu_k} = 0\),

\(C_{\lambda_{c_k}} \geq c\),

\(\log(W_{PC_{\nu_{c_k}}}, W_{ref_{PC_{\nu_{c_k}}}}) = 0\),

\(W_{-1}^{-1} PC_{\nu_{c_k}} = W_{ref_{PC_{\nu_{c_k}}}} = 0\),

\(\mathbf{x} \leq x_k \leq \bar{x}\),

\(\mathbf{u} \leq u_k \leq \bar{u}\),

(37)

where \(q_{\text{nom}}\) defines the robot’s nominal posture, the control inputs are defined by generalized acceleration and contact forces \(u = [a \lambda]\) (i.e., condensed inverse dynamics), \((x, \dot{x}, \ddot{x})\) describes the state trajectory and control sequence, \(N\) is the optimization horizon, the linearized friction cone is defined by \((C, c)\), \(\log(\cdot)\) operator defines the logarithmic manifold needed to handle contact placement that lies on a \(\mathbb{SE}(3)\) manifold, \(W_{PC_{\nu}}, W_{-1}^{-1} W_{PC_{\nu}}\) describes the inverse composition between the reference and current contact placements [70], \((x, \dot{x}, \ddot{x})\) are the state bounds, \((u, \dot{u})\) are the control bounds, \((Q, N, R, K)\) are weighing matrices used in the different cost terms, \(C_k\) and \(G_k\) are the set of active and swing contacts given the kth node. Both contact positions and velocities \((W_{PC_{\nu}}, W_{PC_{\nu}})\) are expressed in the inertial frame \(W\).

1) Inverse dynamics and contact forces: Eq. (37) formulates inverse dynamics as a function of the kinematic integrator \(\mathbf{\psi}\) and nonlinear equality constraints for the condensed inverse dynamics (see Eq. (33)) and zero contact forces in swing feet. Instead, to describe discontinuities when contacts are gained, we employ impulse dynamics during these contact-gain transitions (a term coined by [2]). Regarding their analytical derivatives, Section V-A describes how to compute them for condensed inverse dynamics. Instead, we obtain post-impact velocity and impulse force derivatives (i.e., impulse dynamics derivatives) as explained in [5].

2) Inequalities constraints via quadratic penalty: We define quadratic barrier functions for the friction cone, state and control bounds constraints. This approach shows effective
respectively. hand-tuned weights for the cone, state and control barriers, where

\[ \Phi(x, u) = \sum_{i=0}^{m_{\text{cone}}} \begin{cases} \left[ C \Lambda_c - e \right]_i^2 & \text{if } \left[ C \Lambda_c - e \right]_i \leq 0 \\ 0 & \text{otherwise} \end{cases} \]

\[ \Phi_{\text{state}}(x) = \sum_{i=0}^{n_x} \begin{cases} \left[ x - \bar{x} \right]_i^2 & \text{if } \left[ x - \bar{x} \right]_i \leq 0 \\ 0 & \text{otherwise} \end{cases} \]

\[ \Phi_{\text{control}}(u) = \sum_{i=0}^{n_u} \begin{cases} \left[ u - \bar{u} \right]_i^2 & \text{if } \left[ u - \bar{u} \right]_i \leq 0 \\ 0 & \text{otherwise} \end{cases} \]

where \( \left[ \cdot \right]_i \) is the value of the \( i \)-th element; \( m_{\text{cone}} \) is the number of facets of the friction cone; \( w_{\text{cone}} \), \( w_{\text{state}} \) and \( w_{\text{control}} \) are hand-tuned weights for the cone, state and control barriers, respectively.

3) Soft contact constraints: We formulate contact placement and velocity as soft constraints via quadratic penalties: \( w_{\text{cpos}} \| W_{\text{cpos}} \| \| \| W_{\text{cvel}} \| \| \| \) and \( w_{\text{cvel}} \| W_{\text{cvel}} \| \| \| \) with \( w_{\text{cpos}} \) and \( w_{\text{cvel}} \) as the hand-tuned weights for contact placement and velocity, respectively. Note that we use the \( \ominus \) notation to describe the inverse composition mapped to the tangent space at the identity entity (i.e., a difference operator).

4) Implementation details: The MPC horizon is 1 s, which is described through 100 nodes with timesteps of 10 ms. Each node is numerically integrated using a symplectic Euler scheme. We compensate for delays in communication between the MPC and feedback-policy controller (often around 1 ms), by estimating the initial position and velocity. These initial conditions are defined in our inverse-dynamics MPC. On the other hand, we initialize our algorithm with the previous MPC solution and regularization value \( \mu \). This is to encourage a similar numerical evolution to that seen in the optimal control case. Instead, the penalty parameter \( \nu \) used in the merit function is always updated as in Eq. (27). This is to account for infeasibilities in the initial node and unseen nodes. Note that we refer to unseen nodes as the nodes that appear in each MPC step after receding the horizon.

C. Feedback MPC in joint-torque space

As reported in our previous work [5], classical whole-body controllers (e.g. [71], [11]) do not track angular momentum accurately. This is related to the nonholonomic nature of the kinetic momenta [1], as Brockett’s theorem [72] suggests that these systems cannot be stabilized with continuous time-invariant feedback control laws (i.e., a whole-body controller). In practice, we have observed and reported in [5] that this translates into angular momentum tracking errors. This also influences joint efforts, swing-foot positions and velocities tracking. To avoid these issues, we propose a novel feedback strategy for MPC with inverse dynamics as described below.

Our inverse-dynamics MPC computes control policies for generalized acceleration and contact forces, i.e.,

\[ \delta q^* = \text{ID}(q^*, \nu^*, \delta q^*, \delta \lambda^*), \]

where \( \Pi_a, \Pi_\lambda \) are the feed-forward commands for the reference generalized acceleration \( \Lambda^* \) and contact forces \( \lambda^* \), respectively, and \( \Pi_a, \Pi_\lambda \) are their feedback gains. However, joint-level controllers often track reference joint effort commands, positions and velocities (e.g., [73], [74]).

To map the policy in Eq. (39) into the joint-effort space, we use inverse dynamics, i.e.,

\[ \delta \tau^* = - \Pi \tau - \Pi_\tau \delta x, \]

where \( \Pi_a, \Pi_\lambda \) are the feed-forward commands for the reference generalized acceleration \( \Lambda^* \) and contact forces \( \lambda^* \), respectively, and \( \Pi_a, \Pi_\lambda \) are their feedback gains. However, joint-level controllers often track reference joint effort commands, positions and velocities (e.g., [73], [74]).

To map the policy in Eq. (39) into the joint-effort space, we use inverse dynamics, i.e.,

\[ \delta \tau^* = - \Pi \tau - \Pi_\tau \delta x, \]

where \( \Pi_a, \Pi_\lambda \) are the feed-forward commands for the reference generalized acceleration \( \Lambda^* \) and contact forces \( \lambda^* \), respectively, and \( \Pi_a, \Pi_\lambda \) are their feedback gains.
and feedback. First, we compute the feed-forward joint effort commands $\pi_\tau$ by injecting only the feed-forward terms, i.e.,
\[
\pi_\tau = (\Lambda^*_{\tau})^\dagger \cdot \mathbf{J}(q^*, \pi_a, \pi_\lambda),
\]
\[
= (\Lambda^*_{\tau})^\dagger \cdot (\mathbf{M}(q^*)\pi_a + h(q^*, v^*) - \mathbf{J}_c(q^*)\pi_\lambda),
\]
where $h(q^*, v^*)$ describes the Coriolis and gravitational forces, $\mathbf{M}(q^*)$ the joint-space inertia matrix, and $\mathbf{J}_c(q^*)$ the stack of contact Jacobians evaluated in the optimal robot configuration $q^*$ and velocity $v^*$. Second, we calculate the feedback joint torque gains $\Pi_\tau$ using a similar procedure, i.e., by injecting the feedback terms as follows:
\[
\Pi_\tau = (\Lambda^*_{\tau})^\dagger \cdot (\mathbf{M}(q^*)\Pi_a - \mathbf{J}_c(q^*)\Pi_\lambda).
\]
Thus with Eq. (41) and (42), we compute the joint effort control policy for the duration of the node.

This concludes the description of our OC solver, condensed inverse dynamics, MPC formulation and feedback policy. Below, we provide evidence of the benefits of our method and demonstrate the first application of inverse-dynamics MPC in hardware. Our inverse-dynamics MPC results in state-of-the-art dynamic climbing on the ANYmal robot.

VII. RESULTS

We implemented in C++ both factorizations for the equality-constrained DDP algorithm as well as both inverse-dynamics formulations within the CROCODILYL library [37]. A comparison of our nullspace factorization against the Schur-complement approach highlights the benefits of handling efficiently nonlinear equality constraints (Section VII-B). Results of our feasibility-driven search, merit function and regularization scheme show a broader basin of attraction to local minima (Section VII-C). This enables us to solve nonlinear optimal control problems with poor initialization and generate athletic maneuvers (Section VII-D). Moreover, we compare the numerical effect of condensing the inverse dynamics against the redundant formulation (Section VII-E).

As observed in the literature, our inverse-dynamics approach handles coarse optimization problems better than its forward-dynamics counterpart (Section VII-F). Our approach solves optimal control problems as quickly as the forward-dynamics formulation for a range of different robotics problems (Section VII-G). Indeed, we show that it enables feedback MPC based on inverse dynamics, which is a crucial element to enable the ANYmal robot to climb stairs with missing treads (Section VII-H). Below, we begin by specifying the optimal control problems, PC and solver parameters used in the reported results.

A. Problem specifications and setup

We tested our approach on multiple challenging robotic problems. This allowed us to evaluate the performance of our equality-constrained DDP algorithm with different nonlinear constraints and several decision variables and constraints. The robotic problems are (i) an acrobot moving towards its upward position (pend); (ii) a quadrotor flying through multiple targets (quad); (iii) various quadrupedal gaits: walk (walk), trot-bound (trotb), and pace-jump (pjump) gaits; and (iv) a humanoid reaching a number of grasping points (hum). For the sake of comparison, we use quadratic penalties to encode the soft friction-cone constraints. All these problems include regularization terms for state and control at each node. They also include cost terms that aim to achieve certain desired goals such as position and orientation in the quadrotor or foothold/grasping locations in the legged robots.

We conducted all the experiments on an Intel® Core™ i9-9980HK CPU @ 2.40GHz x 16 running Ubuntu 20.04. We compiled our code with clang++10 utilizing the native architecture flag (i.e., -march=native), which enables the compiler to execute all instructions supported by our CPU. We run our code using 8 threads. We disabled TurboBoost to reduce variability in benchmark timings. We used the following values for the hyper-parameters: $\rho = 0.3$, $\eta_1 = 0.1$, $\eta_2 = 2$, $\alpha_0 = 0.01$, $\alpha_1 = 0.5$, $\kappa_0 = 10^{-4}$, $\beta^1 = 10^6$ and $\beta^d = 10$.

B. Nullspace vs Schur-complement factorizations

Our nullspace factorization always leads to a reduction in computation time (up to 47.3%) when compared to the Schur-complement approach, as shown in Fig. 6. This reduction is higher in problems with a larger number of equality constraints, i.e., the redundant inverse-dynamics formulation with a larger number of contact constraints (walk). The nullspace factorization also reduces computation time per iteration as both factorizations converge within the same number of iterations, cost and accuracy. We obtained the average computation time for each problem at the top of the bar charts.
and QR with column pivoting (null-qr). We used the rank-revealing decompositions available in the Eigen library [19]: LU with full pivoting (Eigen::FullPivLU) and QR with column pivoting (Eigen::ColPivHouseholderQR). It also relies on Eigen’s LU decomposition with partial pivoting (Eigen::PartialPivLU) to compute \((h_u Y)^{-1}\) efficiently, as this matrix is square-invertible. If not indicated otherwise, we use LU with full pivoting decomposition to compute \([Y, Z]\).

C. Benefits of feasibility-driven search

Coldstart or poor initialization affects the convergence of optimal control problems. It is especially relevant in the generation of agile maneuvers such as jumping gaits (e.g., pjump problem). Furthermore, the use of equality constraints to encode the robot’s dynamics (i.e., inverse-dynamics constraints) helps the algorithm’s convergence by balancing optimality and feasibility properly. However, it is also imperative to balance the feasibility of the dynamics (i.e., kinematic evolution) as well, which is ignored in the DDP algorithm. To support this claim, we obtained the percentage of successful resolutions, average cost and iterations with and without dynamics feasibility support.

We computed the average cost from trials that have converged only. In addition, we initialized the algorithm with a random control sequence and a constant state trajectory using the robot’s nominal posture. We used the pace-jump problem (pjump) and performed 500 trials. In Table I, results show that handling dynamics feasibility increases the algorithm’s basin of attraction, convergence rate, and solutions for both inverse-dynamics formulations. In the experiments without dynamic feasibility, we performed the DDP’s forward pass and ignored the gaps in the computation of the expected improvement, i.e., in Eq. (24).

D. Generation of complex maneuvers

Our approach can generate a range of different agile or complex maneuvers within a few iterations. Fig. 7 shows the optimal motions computed by our method for different robotic platforms and tasks. These motions converged between 10 and 50 iterations, except for the gymnastics motions on the Talos robot (Fig. 7a) which took circa 500 iterations. For the problems with the Talos and ANYmal robots (Fig. 7a-c), we provided the sequence of contact constraints and their timings. Furthermore, we did not specify the desired swing motion or encode any heuristic in the generation of the Talos’ motions (Fig. 7a,b). Indeed, its balancing behaviors are derived from the first principles of optimization. It is the same for the obstacle task in Fig. 7d, where a squatting motion is needed to avoid obstacles with the magenta sphere.

E. Redundant vs condensed formulations

We compared the numerical effects of the condensed formulation against the redundant one, as the former assumes that the RNEA constraints are always feasible (refer to Section V-A). In Fig. 8, we can see the normalized cost and \(\ell_1\)-norm feasibility evolution (dynamics and equality constraints) for both inverse-dynamics formulations. There we see that both approaches converged within the same number of iterations and total cost, if feasibility evolved similarly (quad, pjump, trotb and walk). This is contrary to cases where optimality and feasibility did not always decrease monotonically (pend and hum). This trade-off is balanced by our merit function, as defined by Eq. (26). Indeed, redundant formulations increase the possibility that our algorithm balances more effectively both optimality and feasibility. However, this does not represent an overall benefit in most practical cases as the condensed formulations speed up computation time as shown in Fig. 6.

F. Solving coarse optimization problems

We compared the convergence rate using different discretization resolutions. We defined the same cost functions for each problem repetitively but transcribed them with different step integrations. For the sake of simplicity, we employed a symplectic Euler integrator and initialized the algorithms with the robot’s nominal posture and quasi-static inputs at each node of the trajectory. We used the feasibility-driven DDP (FDDP) algorithm [37] provided in CROCODYLYL for the forward-dynamics results. Instead, for the inverse-dynamics cases, we used our equality-constrained DDP algorithm and condensed formulation.

Table II reports the number of iterations required for convergence. These results show that our inverse-dynamics formulation is more likely to converge in very coarse optimizations (e.g., pjump and trotb) and with fewer iterations. Both aspects provide evidence of the benefits of optimal control using inverse dynamics.

<table>
<thead>
<tr>
<th>Problems</th>
<th>100</th>
<th>80</th>
<th>60</th>
<th>50</th>
<th>40</th>
<th>20</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>quad</td>
<td>forward</td>
<td>10</td>
<td>10</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>inverse</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>pjump</td>
<td>forward</td>
<td>20</td>
<td>21</td>
<td>23</td>
<td>24</td>
<td>40</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>inverse</td>
<td>20</td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>19</td>
<td>41</td>
</tr>
<tr>
<td>trotb</td>
<td>forward</td>
<td>15</td>
<td>16</td>
<td>18</td>
<td>25</td>
<td>99</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>inverse</td>
<td>9</td>
<td>9</td>
<td>12</td>
<td>16</td>
<td>36</td>
<td>153</td>
</tr>
<tr>
<td>walk</td>
<td>forward</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>12</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>inverse</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>8</td>
<td>50</td>
<td>14</td>
</tr>
</tbody>
</table>

X algorithm does not find a solution within 200 iterations.
Fig. 7. Snapshots of different maneuvers computed using our inverse-dynamics formulations and equality-constrained DDP. (a) A gymnastic routine performed by the Talos robot. (b) Reaching a few grasping points with the Talos robot. (c) A sequence of walking, trotting and jumping gaits executed by the ANYmal robot. (d) Talos legs squatting to avoid an obstacle. To watch the video, click the picture or see https://youtu.be/NhvSUVopPCI.

Fig. 8. Cost and feasibility evolution for redundant and condensed formulations. (Left) Normalized cost per iteration. (Right) $\ell_1$-norm of the total feasibility (both dynamics and equality constraints) per iteration. A redundant formulation allows our solver to handle the entire problem’s feasibility and optimality, while a condensed one imposes RNEA feasibility. This is key to reducing the total cost of the pend and hum problems.

G. Computation time and convergence rate

We report the total computation time of our approach and the forward-dynamics formulation on six different problems. Fig. 9 shows that solutions with our equality-constrained DDP and condensed formulation are often faster than with the (compact) forward-dynamics formulation proposed in [37]. Our nullspace factorization and condensed formulation are key factors in reducing computation time. The results reported for quadrupedal locomotion (i.e., pjump, trotb, walk) were intentionally designed to mimic realistic MPC setups (around 1.2 s of horizon and 120 nodes) and can run at 50 Hz.

H. MPC and dynamic locomotion

We evaluated the capabilities of our inverse-dynamics MPC and feedback-policy controller to perform dynamic locomotion over complex terrains with the ANYmal robot. Fig. 10 shows relevant instances and ANYmal’s perception when it traverses multiple pallets, beams, and a damaged industrial-like stair. We display three different instances of the ANYmal robot in Fig. 10a and 10b. In each instance, we show the optimal swing-foot trajectories computed by the inverse-dynamics MPC within its prediction horizon. Furthermore, our perception pipeline extracted and updated different convex
Fig. 10. Snapshots of different locomotion maneuvers computed by our inverse-dynamics MPC. All these experimental trials use the feedback-policy controller and onboard state estimation and perception. (a) Walking over a set of pallets of different heights. (b) Crossing a gap between two beams. (c) Climbing up industrial stairs with missing steps. To watch the video, click the picture or see https://youtu.be/NhvSUVopPCI?t=161.
surfaces from a terrain elevation map online. With these surfaces, we select a set of feasible footstep regions to track a velocity command from a joystick.

In Fig. 10a, the height of the first and last pallets is 19 cm. The gap and height difference of the pallet in the middle is 15 cm and 31 cm, respectively. Instead, in the stair-climbing trial reported in Fig. 10c, we removed two treads to emulate a damaged staircase after a disaster. This scaffolding staircase is typically used on construction sites and offshore platforms. Its riser height and tread length are 17 cm and 26 cm, respectively. It has an inclination of 33°. However, when removing a tread, there is a gap of 26 cm in length and 34 cm in height. These are challenging conditions for a robot. Now the equivalent inclination is 37° and the gap distance is around half of the ANYmal robot (approx. 50 cm). Indeed, to the most recent knowledge of the authors, this experimental trial shows ANYmal (or any legged robot) crossing a damaged stair for the first time. It demonstrates the importance of using the robot’s full dynamics, computing the optimal policy, and having a high convergence rate. Finally, we tested our inverse-dynamics MPC in these conditions a few times as reported in the accompanying video: https://youtu.be/NhvSUVopPCI.

I. MPC and feedback policy

Our feedback MPC adapted the robot’s posture to maximize stability and the kinematics needed to transverse the challenging pallets’ heights and the missing treads in the industrial stair. It also tracked accurately the desired footstep placements and swing-foot trajectories. This allowed the ANYmal robot to navigate carefully through narrow regions (beams) and obstacles (stairs). Our previous work [5] also demonstrated that the joint-effort feedback policy computed by the MPC significantly enhanced the tracking of angular momentum, joint efforts, and swing-foot motions. Fig. 11 shows linear and angular momentum tracking when the ANYmal robot crossed the first missed tread in the stair-climbing experiment. Reference/desired signals are blue, while measured signals are red. Our feedback-policy controller accurately tracks the desired momenta.

Fig. 11. Linear (top) and angular (bottom) momentum tracking of our feedback-policy controller when the ANYmal robot crossed the first missed tread in the stair-climbing experiment. Reference/desired signals are blue, while measured signals are red. Our feedback-policy controller accurately tracks the desired momenta.

Fig. 12. Joint torque and force tracking of the ANYmal’s right-front leg when the robot crossed the first missed tread in the stair-climbing experiment. Reference/desired signals are blue, while measured signals are red. Our feedback-policy controller accurately tracks joint torque commands and desired contact forces.

Fig. 13. Comparison between the nullspace and Schur-complement factorizations during a dynamic trotting gait with the ANYmal robot. The Schur-complement factorization is more expensive to compute, and our MPC cannot run at 50 Hz. These delays in the control loop produced instability in the trotting gait (bottom). Instead, the nullspace factorization allowed the MPC to run at 50 Hz and generated a stable trotting gait (top).

J. MPC and nullspace efficiency

We compared the nullspace and Schur-complement factorizations during a dynamic trotting gait with the ANYmal robot. Similarly to Fig. 6, we found that the Schur-complement factorization cannot run our MPC as fast as needed (i.e., 50 Hz). The average frequency was around 38 Hz. This translated into very unstable trotting motions as reported in Fig. 13. Instead, the efficiency of the nullspace factorization allowed us to run the MPC at 50 Hz, which is needed to stabilize this trotting gait.

K. Push recovery and feasibility

We analyzed the capabilities of our inverse-dynamics MPC to reject multiple body disturbances during locomotion. To do
which helps to quickly find a solution (bottom). The bottom plots show the feedback policies to balance the ANYmal robot while walking dynamically in place (top). When this happens, our solver increases the infeasibility instantly, which helps to quickly find a solution (bottom). The bottom plots show the \( \ell_1 \)-norm of the kinematics \(|f_x|\) and dynamics \(|h_u|\) constraints using a logarithm scale.

so, we evaluated the feasibility evolution of the kinematics and inverse-dynamics (equality) constraints. We found that when we pushed the ANYmal robot (Fig. 14), our equality-constrained DDP algorithm increased the infeasibility (especially for the inverse-dynamics constraints). This is because our merit function balances optimality and feasibility in real time. However, these increments in the infeasibility of the current solution are quickly reduced in the next MPC iterations. While this might appear to be a disadvantage, it is an element that keeps the solver iterating around a desired push-recovery behavior, which increases the convergence rate and locomotion robustness.

VIII. CONCLUSION

In this paper, we propose a novel equality-constrained DDP algorithm for solving optimal control problems with inverse dynamics efficiently. Our nullspace-parametrization approach leads to a backward pass\(^1\) in which its computational cost does not grow with respect to the number of equality constraints. Quite the opposite, it reduces the dimensionality of the matrix to be decomposed using Cholesky in the Riccati recursion (a serial computation). This is possible as we can partially parallelize some of the computations needed in the Riccati recursion. We provided evidence that our method reduces computational time (up to 47.3\%) and can solve coarse optimal control problems with different robots and constraints (up to 10 Hz of trajectory discretization). Despite that computing the image and kernel \([Y, Z]\) increases algorithm complexity, it reduces overall computation times due to the possibility of parallelization.

Our algorithm is designed to drive both dynamics and equality-constraint feasibility towards zero. To achieve this, it employs a Goldstein-inspired condition based on a feasibility-aware expected improvement and merit function. This increases the robustness against poor initialization, enables the generation of agile and complex maneuvers, and allows our predictive controller to adapt quickly to body disturbances.

To further improve computation efficiency, we presented a condensed formulation of inverse dynamics that can handle arbitrary actuation models. To do so, we assumed that the inverse-dynamics constraint is always feasible and defined an under-actuation constraint. This assumption was validated empirically, and we found that most of the time this assumption had no impact on numerical behavior.

Our approach enables the generation of complex maneuvers as fast as needed for MPC applications. Indeed, we presented the first application of an inverse-dynamics MPC to a legged robot. We also proposed a novel approach to compute feedback policy in the joint-effort space, which allows us to directly control the robot. Within a perceptive locomotion pipeline, both the inverse-dynamics MPC and feedback policy enabled the ANYmal robot to navigate over complex terrains such as damaged staircases.

APPENDIX A

We can break the optimal control problem into smaller sub-problems as the following relationship holds for each node:

\[
\xi_k^+ = \nu_{x_{k+1}} + \nu_{xx_{k+1}} \delta x_{k+1} \quad \forall k = \{N, N-1, \ldots, 1\}
\]

We know that the KKT problem at the terminal node is defined as

\[
[-I \quad L_{xx_N}] \begin{bmatrix} \xi_N^+ \n \delta x_N \end{bmatrix} = -\ell_{x_N}.
\]

By definition, we have \( \nu_{x_N} = \ell_{x_N} \) and \( \nu_{xx_N} = L_{xx_N} \), which both represent the relationship above at the terminal node. Without sacrificing generality, we compute the remaining derivatives of the value function \( \nu_{x_k} \) and \( \nu_{xx_k} \) recursively by solving the next sub-problem, i.e.,

\[
\begin{bmatrix} -I & L_{xx_k} & L_{ux_k}^T & f_{x_k}^T & f_{u_k}^T & -I \\ L_{ux_k} & L_{uu_k} & f_{u_k} & -I & \nu_{xx_k} & \nu_{xxk} \end{bmatrix} \begin{bmatrix} \xi_k^+ \\
\delta x_k \\
\delta u_k \\
\xi_{k-1} \\
\delta x_{k-1} \\
\delta u_{k-1} \end{bmatrix} = \begin{bmatrix} \ell_{x_k} \\
\ell_{u_k} \\
\bar{f}_k \\
\nu_{x_{k-1}} \\
\nu_{xx_{k-1}} \end{bmatrix}
\]

Then, condensing the third and fourth rows yields

\[
\begin{bmatrix} -I & Q_{xx_k} & Q_{ux_k} & Q_{uu_k} \\ Q_{xx_k} & Q_{xxk} & Q_{uxk} & Q_{uuk} \end{bmatrix} \begin{bmatrix} \xi_k^+ \\
\delta x_k \\
\delta u_k \\
\xi_{k-1} \\
\delta x_{k-1} \\
\delta u_{k-1} \end{bmatrix} = \begin{bmatrix} Q_{xx_k} \\
Q_{xxk} \\
Q_{uxk} \\
Q_{uuk} \end{bmatrix}
\]

with

\[
Q_{xx_k} = \ell_{x_k} + f_{x_k}^T (\nu_{x_{k+1}} + \nu_{xx_{k+1}} \bar{f}_k),
\]

\[
Q_{ux_k} = \ell_{u_k} + f_{u_k}^T (\nu_{x_{k+1}} + \nu_{xx_{k+1}} \bar{f}_k),
\]

\[
Q_{xxk} = L_{xx_k} + f_{x_k}^T (\nu_{xxk} + \nu_{xxk} f_{x_k}),
\]

\[
Q_{uxk} = L_{ux_k} + f_{u_k}^T (\nu_{xxk} + \nu_{xxk} f_{x_k}),
\]

\[
Q_{uuk} = L_{uu_k} + f_{u_k}^T (\nu_{xxk} + \nu_{xxk} f_{u_k}).
\]

This can be further condensed by injecting the optimal policy

\[
\delta u_k = -\pi_k - \Pi_k \delta x_k
\]

\(^1\)The backward pass is the routine that computes the search direction (see Algorithm 1).
into the previous equation. This leads to the following relationship for the next costate at node \( k - 1 \):
\[
\xi_{k-1} = \mathcal{V}_{\xi_k} + \mathcal{V}_{\xi_{k+1}} \delta x_k
\]
with
\[
\mathcal{V}_{\xi_k} = Q_{\xi_k} + \Pi_k^T (Q_{uu_k} \pi_k - Q_{uu_k} - 2Q^T_{\xi u_k} \Pi_k),
\]
\[
\mathcal{V}_{\xi_{k+1}} = Q_{\xi_{k+1}} + (\Pi_{k+1} Q_{uu_k} - 2Q^T_{\xi u_{k+1}} \Pi_k).
\]
We then inject this costate relationship to solve each subproblem backwards.

**Author contributions**

Carlos Mastalli devised the main ideas behind nullspace parametization, condensed inverse-dynamics, MPC formulation, and feedback control, and took the lead in writing the manuscript and preparing the video. Saroj Prasad Chhatoi developed the Schur-complement approach and the redundant formulation. Thomas Corbère integrated the inverse-dynamics MPC into a perceptive locomotion pipeline, developed its main components, and supported the experimental trials on the ANYmal robot. Sethu Vijayakumar and Steve Tonneau provided critical feedback and helped shape the manuscript.

**REFERENCES**


H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning...
L. zhi Liao and C. A. Shoemaker, "Advantages of Differential Dynamic
R. Deits and R. Tedrake, "Footstep planning on uneven terrain with
G. H. Golub and C. F. V. Loan, Matrix computations
W. Jallet, J. Carpentier, and N. Mansard, "Implicit Differential Dynamic
R. Fletcher, "A modified Marquardt subroutine for non-linear least
J. Nocedal and S. Wright, Numerical Optimization, 2nd ed. New York,
R. H. Byrd, M. E. Hribar, and J. Nocedal, "An Interior Point Algorithm
R. H. Byrd, R. Himmelblau, and M. Marquardt, "A quasi-Newton algorithm for
C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Hi-

Carlos Mastalli received an M.Sc. degree in mechatronics engineering from the Simon Bolivar University, Caracas, Venezuela, in 2013 and a Ph.D. degree in bio-engineering and robotics from the Istituto Italiano di Tecnologia, Genoa, Italy, in 2017.
He is currently an Assistant Professor at Heriot-Watt University, Edinburgh, U.K. He is the Head of the Robot Motor Intelligence (RoMI) Lab affiliated with the National Robotics and Edinburgh Centre for Robotics. He is also appointed as Research Scientist at IHMC, USA. Previously, he conducted cutting-edge research in several world-leading labs: Istituto Italiano di Tecnologia (Italy), LAAS-CNRS (France), ETH Zurich (Switzerland), and the University of Edinburgh (UK). His research focuses on building athletic robots with legs and arms. Carlos’ research work is at the intersection of model predictive control, numerical optimization, machine learning, and robot co-design.

Saroj Prasad Chhatoi received his M.Sc. degree in physics from the Institute of Mathematical Sciences, Chennai, India in 2019.
He is currently pursuing a Ph.D. degree in Information Engineering at Centro de Ricerca “Enrico Piaggio” Theoretical Physics, Pisa, Italy. His research interests include legged locomotion, control of soft robotic systems, model predictive control.

Thomas Corbères received his M.Sc. degree in informatics and computing engineering from the University of Toulouse, Toulouse, France in 2020.
He is currently pursuing a Ph.D. degree in robotics and autonomous systems at the University of Edinburgh under the supervision of S. Tonneau. His research interests include legged locomotion, model predictive control, and contact planning.

Steve Tonneau received the Ph.D. degree in humanoid robotics from the INRIA/IRISA, France, in 2015.
He is a Lecturer at the University of Edinburgh, Edinburgh, U.K. Previously, he was a Postdoctoral Researcher at LAAS-CNRS in Toulouse, France. His research focuses on motion planning based on the biomechanical analysis of motion invariants. Applications include computer graphics animation as well as robotics.
Sethu Vijayakumar received the Ph.D. degree in computer science and engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1998. He is Professor of Robotics and Founding Director of the Edinburgh Centre for Robotics, where he holds the Royal Academy of Engineering Microsoft Research Chair in Learning Robotics within the School of Informatics at the University of Edinburgh, U.K. He also has additional appointments as an Adjunct Faculty with the University of Southern California, Los Angeles, CA, USA and a Visiting Research Scientist with the RIKEN Brain Science Institute, Tokyo. His research interests include statistical machine learning, whole body motion planning and optimal control in robotics, optimization in autonomous systems as well as optimality in human motor motor control and prosthetics and exoskeletons. Professor Vijayakumar is a Fellow of the Royal Society of Edinburgh. In his recent role as the Programme Director for Artificial Intelligence and Robotics at The Alan Turing Institute, Sethu helps shape and drive the UK national agenda in Robotics and Autonomous Systems.