Inverse-Dynamics MPC via Nullspace Resolution

Citation for published version:
Mastalli, C, Chhatoi, SP, Corbéres, T, Tonneau, S & Vijayakumar, S 2023, 'Inverse-Dynamics MPC via Nullspace Resolution', *IEEE Transactions on Robotics*, vol. 39, no. 4, 10101674, pp. 3222-3241.
https://doi.org/10.1109/TRO.2023.3262186

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

Published In:
IEEE Transactions on Robotics

Inverse-Dynamics MPC via Nullspace Resolution

Carlos Mastalli† Saroj Prasad Chhatoi† Thomas Corbères Steve Tonneau Sethu Vijayakumar

***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 CROCODDYL.**

***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,

This research was supported by (1) the European Commission under the Horizon 2020 project Memory of Motion (MEMMO, project ID: 780684), Natural Intelligence (project ID: 101016970) (2) the Engineering and Physical Sciences Research Council (EPSRC) UK RAI Hub for Offshore Robotics for Certification of Assets (ORCA, grant reference EP/R026173/1), and (3) the Alan Turing Institute. †These are the leading authors of this work.
(Corresponding author: 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).

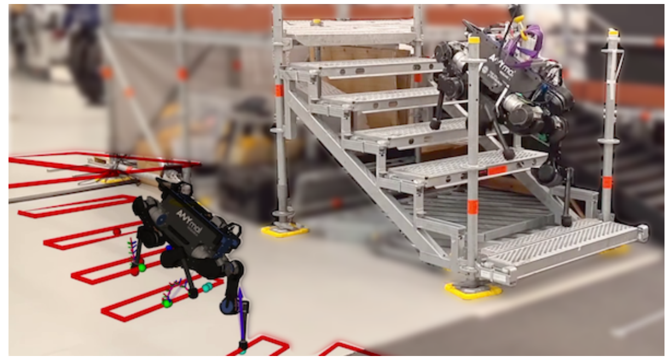

Fig. 1. ANYmal climbs up a damaged staircase using our inverse-dynamics MPC via our nullspace parametrization. In the instance depicted in this figure, ANYmal is crossing a gap incurred by a missing tread. The gap dimension is 26 cm in length and 34 cm in height, which represents an inclination of 37° and around half of the ANYmal robot. The footstep 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/NhvSUVopPCI`.

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 advantages in MPC for legged robots helps to generate complex maneuvers such as the ones needed to climb up a damaged staircase (Fig. 1).

The DDP algorithm exploits the *structure* of optimal control problems based on *forward models*. This is a critical aspect of the deployment of model predictive controllers in legged robots, as we can see in recent works [3], [4], [5], [12]. However, in contrast to forward-dynamics formulations, there are no algorithms available to do the same based on *inverse models*, i.e., algorithms that exploit the structure of inverse-dynamics formulations. This is a limiting factor to deploying *inverse-dynamics MPC* in legged robots. To address this issue, we propose an approach that takes advantage of the structure of optimal control problems with inverse-dynamics constraints. We summarize our method as follows. First, we parametrize the inverse-dynamics constraints using its nullspace basis to exploit the temporal structure. This allows us to perform parallel computations that reduce the computational cost of solving OC problems. Second, we condense the inverse dynamics and inject its sparsity into the computation of the action-value function to exploit the functional structure.

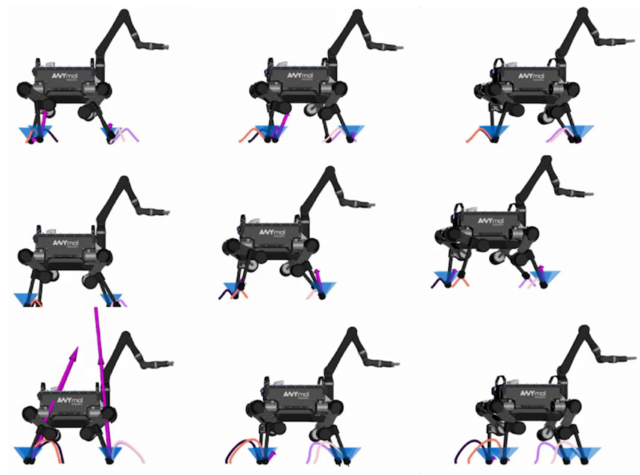

Fig. 2. Illustration of various locomotion gaits optimized using our inverse-dynamics formulation and solver for ANYmal with a Kinova arm. These motions were optimized in less than 20 iterations and 500 milliseconds. (Top) Multiple walking gaits with 25 cm strides. (Middle) Several trotting gaits with 10 cm strides. (Bottom) Multiple jumping gaits of 30 cm in length. To watch the video, click the figure or see `https://youtu.be/NhvSUVopPCI?t=10`.

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 CROCODDYL. 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 an 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 INTERIOR/RICCATI. Both INTERIOR/DIRECT and INTERIOR/RICCATI techniques handle inequality constraints using primal-dual interior point and line search. However, INTERIOR/RICCATI is significantly faster (at least one order of magnitude) than INTERIOR/DIRECT. This is because the Riccati recursions exploit the problem's temporal structure. Nevertheless, 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], INTERIOR/DIRECT often struggles to get solutions with very low constraint satisfaction (i.e., lower than 10^{-9}) 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.

B. 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 *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 *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.

III. 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.

A. Optimal control with the forward dynamics

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

$$\begin{aligned}
\min_{(\hat{\mathbf{q}}_s,\hat{\mathbf{v}}_s),(\hat{\boldsymbol{\tau}}_s,\hat{\boldsymbol{\lambda}}_s)} \;& \ell_N(\mathbf{q}_N,\mathbf{v}_N) + \sum_{k=0}^{N-1}\int_{t_k}^{t_{k+1}} \ell_k(\mathbf{q}_k,\mathbf{v}_k,\boldsymbol{\tau}_k,\boldsymbol{\lambda}_k)\,dt \\
\text{s.t.}\;\; & [\mathbf{q}_{k+1},\mathbf{v}_{k+1}] = \boldsymbol{\psi}(\mathbf{q}_k,\mathbf{v}_k,\mathrm{FD}(\mathbf{q}_k,\mathbf{v}_k,\boldsymbol{\tau}_k,\boldsymbol{\lambda}_k)), \\
& \mathbf{q}_k\in\mathcal{Q},\; \mathbf{v}_k\in\mathcal{V},\; \boldsymbol{\tau}_k\in\mathcal{P},\; \boldsymbol{\lambda}_k\in\mathcal{F},
\end{aligned} \tag{1}$$

where $\mathbf{q}_k$, $\mathbf{v}_k$, $\boldsymbol{\tau}_k$, and $\boldsymbol{\lambda}_k$ 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; $\hat{\mathbf{q}}_s$, $\hat{\mathbf{v}}_s$, $\hat{\boldsymbol{\tau}}_s$ and $\hat{\boldsymbol{\lambda}}_s$ 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; $\boldsymbol{\psi}(\cdot)$ defines the integrator function; $\mathrm{FD}(\cdot)$ represents the forward dynamics, which computes the generalized accelerations $\dot{\mathbf{v}}$ through the articulate body algorithm (ABA) [2]:

$$\mathbf{M}\dot{\mathbf{v}} = \boldsymbol{\tau}_{bias} + \mathbf{J}_c^\top\boldsymbol{\lambda}, \tag{2}$$

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

$$\begin{bmatrix}\dot{\mathbf{v}}\\ -\boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\mathbf{M} & \mathbf{J}_c^\top\\ \mathbf{J}_c & \mathbf{0}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{\tau}_{bias}\\ -\mathbf{a}_c\end{bmatrix}. \tag{3}$$

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

B. 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:

$$\begin{aligned}
&[\mathbf{q}_{k+1},\mathbf{v}_{k+1}] = \boldsymbol{\psi}(\mathbf{q}_k,\mathbf{v}_k,\mathbf{a}_k), \\
&\mathrm{ID}(\mathbf{q}_k,\mathbf{v}_k,\mathbf{a}_k,\boldsymbol{\tau}_k,\boldsymbol{\lambda}_k) = \mathbf{0},
\end{aligned} \tag{4}$$

where $\mathrm{ID}(\cdot)$ is the *inverse-dynamics function*. This function is typically computed through the recursive Newton-Euler algorithm (RNEA) [2] if we do not consider the contact constraints $\mathbf{J}_c\dot{\mathbf{v}}-\mathbf{a}_c=\mathbf{0}$. This formulation can be interpreted as a *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 kth node are now $\mathbf{q}_k$, $\mathbf{v}_k$, $\mathbf{a}_k$, $\boldsymbol{\tau}_k$, and $\boldsymbol{\lambda}_k$, 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 contact forces or generalized accelerations 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 forward 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

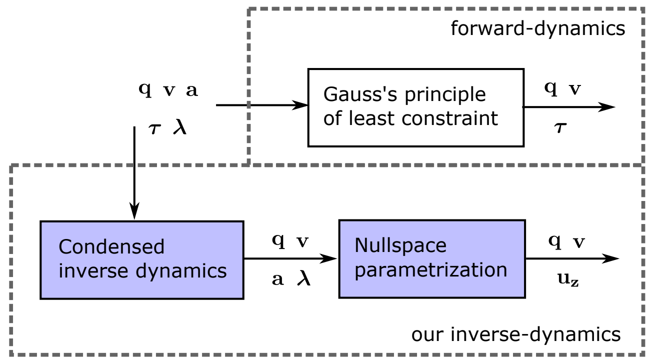

Fig. 3. Comparison of optimal control using forward dynamics and our inverse dynamic approach. In forward-dynamics settings, we can apply Gauss's principle of least constraint to reduce the number of decision variables, as we have a compact representation that depends on $\mathbf{q}, \mathbf{v}$, and $\boldsymbol{\tau}$ only. In contrast, this principle cannot be applied to inverse-dynamics models to reduce the number of decision variables. Instead, our inverse-dynamics approach can be interpreted as a reduction in system dimensionality by first condensing the dynamics and then using nullspace parametrization. Note that $\mathbf{u_z}$ is a representation of $(\mathbf{a}, \boldsymbol{\lambda})$ that comes from nullspace parametrization. Section IV-B and V-A provide a rigorous description of nullspace parametrization and condensed inverse dynamics, respectively.

i.e., stagewise. Here, we are interested in solving the following optimal control problem efficiently:

$$\begin{aligned} &\min_{\hat{\mathbf{x}}_s, \hat{\mathbf{u}}_s} \ \ell_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} \ell_k(\mathbf{x}_k, \mathbf{u}_k) \\ &\text{s.t.} \ \ \mathbf{x}_0 = \tilde{\mathbf{x}}_0, \quad \mathbf{x}_{k+1} = \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k), \quad \mathbf{h}_k(\mathbf{x}_k, \mathbf{u}_k) = \mathbf{0}, \end{aligned} \tag{5}$$

where, similarly to Eq. (4), $\mathbf{f}(\cdot)$ defines the *kinematic evolution* of the system (i.e., integrator), $\mathbf{h}(\cdot)$ ensures the feasibility of the dynamics through its inverse-dynamics function, $\tilde{\mathbf{x}}_0$ represents the initial condition of the system, $\mathbf{x} := (\mathbf{q}, \mathbf{v}) \in \mathcal{X}$ is the state of the system and lies in a differential manifold (with dimension n_x), and $\mathbf{u} \in \mathbb{R}^{n_u}$ is the input of the system that depends on the inverse-dynamics formulation as described in Section V. Note that $\hat{\mathbf{x}}_s$ and $\hat{\mathbf{u}}_s$ 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.,

$$\begin{aligned}
&\nabla_{\hat{\mathbf{w}}_s}\mathcal{L}(\hat{\mathbf{w}}_s,\hat{\boldsymbol{\gamma}}_s,\hat{\boldsymbol{\xi}}_s)=\mathbf{0}, && \text{(stationary condition)} \quad (6)\\
&\tilde{\mathbf{x}}_0\ominus\mathbf{x}_0=\mathbf{0}, && \text{(primal feasibility)}\\
&\mathbf{h}_k(\mathbf{x}_k,\mathbf{u}_k)=\mathbf{0}, && \forall k=\{0,\cdots,N-1\}\\
&\mathbf{f}_k(\mathbf{x}_k,\mathbf{u}_k)\ominus\mathbf{x}_{k+1}=\mathbf{0}, && \forall k=\{0,\cdots,N-1\} \quad (7)
\end{aligned}$$

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

$$\mathcal{L}(\hat{\mathbf{w}}_s,\hat{\boldsymbol{\gamma}}_s,\hat{\boldsymbol{\xi}}_s)=\hat{\mathbf{c}}_s(\hat{\mathbf{w}}_s)+\hat{\boldsymbol{\gamma}}_s^\top\hat{\mathbf{h}}_s(\hat{\mathbf{w}}_s)+\hat{\boldsymbol{\xi}}_s^\top\hat{\mathbf{d}}_s(\hat{\mathbf{w}}_s), \quad (8)$$

with the above terms as

$$\begin{aligned}
\hat{\mathbf{c}}_s(\hat{\mathbf{w}}_s)&=\ell_N(\mathbf{x}_N)+\sum_{k=0}^{N-1}\ell_k(\mathbf{x}_k,\mathbf{u}_k),\\
\hat{\boldsymbol{\gamma}}_s^\top\hat{\mathbf{h}}_s(\hat{\mathbf{w}}_s)&=\sum_{k=0}^{N-1}\hat{\boldsymbol{\gamma}}_k^\top\mathbf{h}_k(\mathbf{x}_k,\mathbf{u}_k),\\
\hat{\boldsymbol{\xi}}_s^\top\hat{\mathbf{d}}_s(\hat{\mathbf{w}}_s)&=\boldsymbol{\xi}_0^\top(\tilde{\mathbf{x}}_0\ominus\mathbf{x}_0)+\\
&\qquad\sum_{k=0}^{N-1}\boldsymbol{\xi}_{k+1}^\top(\mathbf{f}_k(\mathbf{x}_k,\mathbf{u}_k)\ominus\mathbf{x}_{k+1}).
\end{aligned}$$

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

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

$$\begin{bmatrix}\nabla^2_{\hat{\mathbf{w}}_s}\mathcal{L} & \nabla\hat{\mathbf{h}}_s^\top & \nabla\hat{\mathbf{d}}_s^\top\\ \nabla\hat{\mathbf{h}}_s & & \\ \nabla\hat{\mathbf{d}}_s & & \end{bmatrix}\begin{bmatrix}\delta\hat{\mathbf{w}}_s\\ \hat{\boldsymbol{\gamma}}_s^+\\ \hat{\boldsymbol{\xi}}_s^+\end{bmatrix}=-\begin{bmatrix}\nabla\hat{\mathbf{c}}_s\\ \hat{\mathbf{h}}_s\\ \hat{\mathbf{d}}_s\end{bmatrix}, \quad (9)$$

where $\delta\hat{\mathbf{w}}_s$ is the search direction computed for the primal variables, $\hat{\boldsymbol{\gamma}}_s^+:=\hat{\boldsymbol{\gamma}}_s+\delta\hat{\boldsymbol{\gamma}}_s$ and $\hat{\boldsymbol{\xi}}_s^+:=\hat{\boldsymbol{\xi}}_s+\delta\hat{\boldsymbol{\xi}}_s$ 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:

$$\begin{bmatrix}\mathcal{L}_{\mathbf{xx}} & \mathcal{L}_{\mathbf{ux}}^\top & \mathbf{h}_{\mathbf{x}}^\top & \mathbf{f}_{\mathbf{x}}^\top & \\ \mathcal{L}_{\mathbf{ux}} & \mathcal{L}_{\mathbf{uu}} & \mathbf{h}_{\mathbf{u}}^\top & \mathbf{f}_{\mathbf{u}}^\top & \\ \mathbf{h}_{\mathbf{x}} & \mathbf{h}_{\mathbf{u}} & & & \\ \mathbf{f}_{\mathbf{x}} & \mathbf{f}_{\mathbf{u}} & & & -\mathbf{I}\\ & & & -\mathbf{I} & \mathcal{V}'_{\mathbf{xx}}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\\ \boldsymbol{\gamma}^+\\ \boldsymbol{\xi}^+\\ \delta\mathbf{x}'\end{bmatrix}=-\begin{bmatrix}\boldsymbol{\ell}_{\mathbf{x}}\\ \boldsymbol{\ell}_{\mathbf{u}}\\ \bar{\mathbf{h}}\\ \bar{\mathbf{f}}\\ \mathcal{V}'_{\mathbf{x}}\end{bmatrix}, \quad (10)$$

with

$$\begin{aligned}
&\mathcal{L}_{\mathbf{xx}}:=\boldsymbol{\ell}_{\mathbf{xx}}+\mathcal{V}'_{\mathbf{x}}\cdot\mathbf{f}_{\mathbf{xx}}, && \mathcal{L}_{\mathbf{xu}}:=\boldsymbol{\ell}_{\mathbf{xu}}+\mathcal{V}'_{\mathbf{x}}\cdot\mathbf{f}_{\mathbf{xu}},\\
&\mathcal{L}_{\mathbf{uu}}:=\boldsymbol{\ell}_{\mathbf{uu}}+\mathcal{V}'_{\mathbf{x}}\cdot\mathbf{f}_{\mathbf{uu}}, && \boldsymbol{\gamma}^+:=\boldsymbol{\gamma}+\delta\boldsymbol{\gamma},\\
&\boldsymbol{\xi}^+:=\boldsymbol{\xi}+\delta\boldsymbol{\xi}, && \bar{\mathbf{h}}:=\mathbf{h}(\mathbf{x},\mathbf{u}),\\
&\bar{\mathbf{f}}:=\mathbf{f}(\mathbf{x},\mathbf{u})\ominus\mathbf{x}',
\end{aligned}$$

where $\boldsymbol{\ell}_{\mathbf{p}}$, $\mathbf{h}_{\mathbf{p}}$, $\mathbf{f}_{\mathbf{p}}$ correspond to the first derivatives of the cost, equality constraint, and system dynamics with respect to $\mathbf{p}$, respectively; $\mathbf{f}_{\mathbf{pp}}$ is the second derivative of the system dynamics; $\mathcal{V}'_{\mathbf{x}}$, $\mathcal{V}'_{\mathbf{xx}}$ are the gradient and Hessian of the value function; $\boldsymbol{\gamma}$, $\boldsymbol{\xi}$ are the Lagrange multipliers associated with the equality constraint and system dynamics; $\bar{\mathbf{h}}$, $\bar{\mathbf{f}}$ describe the gaps in the equality constraint and dynamics; $\delta\mathbf{x}$, $\delta\mathbf{u}$, $\delta\mathbf{x}'$ and $\delta\boldsymbol{\gamma}$, $\delta\boldsymbol{\xi}$ provides the search direction computed for the primal and dual variables, respectively; $\mathcal{V}'_{\mathbf{x}}\cdot\mathbf{f}_{\mathbf{pp}}$ defines the tensor product with the Hessian of the system dynamics. Finally, note that (i) $\mathbf{p}$ is a hypothetical decision variable that represents $\mathbf{x}$ or $\mathbf{u}$, (ii) we have dropped the node index k and introduced the $'$ notation to describe the node index $k+1$, and (iii) $\bar{\mathbf{f}}$ corresponds to the *kinematic gap* while $\bar{\mathbf{h}}$ 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

$$\boldsymbol{\xi}^+=\mathcal{V}'_{\mathbf{x}}+\mathcal{V}'_{\mathbf{xx}}\delta\mathbf{x}' \quad (11)$$

as shown in Appendix A. This equation connects the derivatives of the value function with the *next costate* $\boldsymbol{\xi}^+$. 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

$$\begin{bmatrix}\mathbf{Q}_{\mathbf{xx}} & \mathbf{Q}_{\mathbf{ux}}^\top & \mathbf{h}_{\mathbf{x}}^\top\\ \mathbf{Q}_{\mathbf{ux}} & \mathbf{Q}_{\mathbf{uu}} & \mathbf{h}_{\mathbf{u}}^\top\\ \mathbf{h}_{\mathbf{x}} & \mathbf{h}_{\mathbf{u}} & \end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\\ \boldsymbol{\gamma}^+\end{bmatrix}=-\begin{bmatrix}\mathbf{Q}_{\mathbf{x}}\\ \mathbf{Q}_{\mathbf{u}}\\ \bar{\mathbf{h}}\end{bmatrix}, \quad (12)$$

where the $\mathbf{Q}$'s describe the local approximation of the *action-value function* in the free space:

$$\begin{aligned}\mathbf{Q_x} &= \boldsymbol{\ell}_\mathbf{x} + \mathbf{f}_\mathbf{x}^\top \mathcal{V}_\mathbf{x}^+, &\quad \mathbf{Q_u} &= \boldsymbol{\ell}_\mathbf{u} + \mathbf{f}_\mathbf{u}^\top \mathcal{V}_\mathbf{x}^+,\\ \mathbf{Q_{xx}} &= \mathcal{L}_\mathbf{xx} + \mathbf{f}_\mathbf{x}^\top \mathcal{V}'_\mathbf{xx}\mathbf{f}_\mathbf{x}, &\quad \mathbf{Q_{ux}} &= \mathcal{L}_\mathbf{ux} + \mathbf{f}_\mathbf{u}^\top \mathcal{V}'_\mathbf{xx}\mathbf{f}_\mathbf{x},\\ \mathbf{Q_{uu}} &= \mathcal{L}_\mathbf{uu} + \mathbf{f}_\mathbf{u}^\top \mathcal{V}'_\mathbf{xx}\mathbf{f}_\mathbf{u}, \end{aligned} \tag{13}$$

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

$$\begin{aligned}\Delta\mathcal{V} = \min_{\delta\mathbf{u}}\ & \frac{1}{2}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}^\top \begin{bmatrix}\mathbf{Q_{xx}} & \mathbf{Q_{ux}}^\top\\ \mathbf{Q_{ux}} & \mathbf{Q_{uu}}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix} + \begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix}^\top\begin{bmatrix}\mathbf{Q_x}\\ \mathbf{Q_u}\end{bmatrix}\\ \text{s.t.}\ & \begin{bmatrix}\mathbf{h_x} & \mathbf{h_u}\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}\\ \delta\mathbf{u}\end{bmatrix} + \bar{\mathbf{h}} = \mathbf{0},\end{aligned} \tag{14}$$

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

$$\begin{bmatrix}\mathbf{Q_{uu}} & \mathbf{h_u}^\top\\ \mathbf{h_u} & \end{bmatrix}\begin{bmatrix}\delta\mathbf{u}\\ \boldsymbol{\gamma}^+\end{bmatrix} = -\begin{bmatrix}\mathbf{Q_u} + \mathbf{Q_{ux}}\delta\mathbf{x}\\ \bar{\mathbf{h}} + \mathbf{h_x}\delta\mathbf{x}\end{bmatrix}. \tag{15}$$

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 $\delta\mathbf{u}$) by factorizing Eq. (15) through the *Schur-complement* approach:

$$\begin{aligned}\delta\mathbf{u} &= -\mathbf{k} - \mathbf{K}\delta\mathbf{x} - \boldsymbol{\Psi}_s\boldsymbol{\gamma}^+,\\ \boldsymbol{\gamma}^+ &:= \tilde{\mathbf{Q}}_\mathbf{uu}(\mathbf{k}_s + \mathbf{K}_s\,\delta\mathbf{x}),\end{aligned} \tag{16}$$

where $\mathbf{k} = \mathbf{Q_{uu}^{-1}}\mathbf{Q_u}$, $\mathbf{K} = \mathbf{Q_{uu}^{-1}}\mathbf{Q_{ux}}$ are the feed-forward and feedback gain on the free space, $\boldsymbol{\Psi}_s = \mathbf{Q_{uu}^{-1}}\mathbf{h_u}^\top$, $\tilde{\mathbf{Q}}_\mathbf{uu} = (\mathbf{h_u}\mathbf{Q_{uu}^{-1}}\mathbf{h_u}^\top)^{-1}$, $\mathbf{k}_s = \bar{\mathbf{h}} - \mathbf{h_u}\mathbf{k}$, $\mathbf{K}_s = \mathbf{h_x} - \mathbf{h_u}\mathbf{K}$ 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

$$\delta\mathbf{u} = -\boldsymbol{\pi}_s - \boldsymbol{\Pi}_s\,\delta\mathbf{x},$$

with

$$\begin{aligned}\boldsymbol{\pi}_s &:= \mathbf{k} + (\mathbf{k}_s^\top\tilde{\mathbf{Q}}_\mathbf{uu}\boldsymbol{\Psi}_s^\top)^\top \quad \text{(feed-forward)},\\ \boldsymbol{\Pi}_s &:= \mathbf{K} + (\mathbf{K}_s^\top\tilde{\mathbf{Q}}_\mathbf{uu}\boldsymbol{\Psi}_s^\top)^\top \quad \text{(feedback gain)}.\end{aligned} \tag{17}$$

This factorization technique requires performing two Cholesky decompositions for computing $\mathbf{Q_{uu}^{-1}}$ and $\tilde{\mathbf{Q}}_\mathbf{uu}$. 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 $\tilde{\mathbf{Q}}_\mathbf{uu}$ 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 $\mathbf{h_u}$, we can decompose/parametrize the decision variable $\delta\mathbf{u}$ as follows

$$\delta\mathbf{u} = \mathbf{Y}\delta\mathbf{u_y} + \mathbf{Z}\delta\mathbf{u_z}, \tag{18}$$

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

$$\begin{bmatrix}\mathbf{Z}^\top\mathbf{Q_{uu}}\mathbf{Z} & \mathbf{Z}^\top\mathbf{Q_{uu}}\mathbf{Y}\\ & \mathbf{h_u}\mathbf{Y}\end{bmatrix}\begin{bmatrix}\delta\mathbf{u_z}\\ \delta\mathbf{u_y}\end{bmatrix} = -\begin{bmatrix}\mathbf{Z}^\top(\mathbf{Q_u} + \mathbf{Q_{ux}}\delta\mathbf{x})\\ \bar{\mathbf{h}} + \mathbf{h_x}\delta\mathbf{x}\end{bmatrix}, \tag{19}$$

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

$$\delta\mathbf{u} = -\boldsymbol{\pi}_n - \boldsymbol{\Pi}_n\,\delta\mathbf{x},$$

with

$$\begin{aligned}\boldsymbol{\pi}_n &:= \mathbf{Z}\mathbf{k}_n + \tilde{\mathbf{Q}}_\mathbf{zz}\boldsymbol{\Psi}_n\bar{\mathbf{h}} \quad \text{(feed-forward)},\\ \boldsymbol{\Pi}_n &:= \mathbf{Z}\mathbf{K}_n + \tilde{\mathbf{Q}}_\mathbf{zz}\boldsymbol{\Psi}_n\mathbf{h_x} \quad \text{(feedback gain)},\end{aligned} \tag{20}$$

where $\mathbf{k}_n = \mathbf{Q_{zz}^{-1}}\mathbf{Q_z}$, $\mathbf{K}_n = \mathbf{Q_{zz}^{-1}}\mathbf{Q_{zx}}$ are the feed-forward and feedback gain associated with the nullspace of the equality constraint, $\tilde{\mathbf{Q}}_\mathbf{zz} = \mathbf{I} - \mathbf{Z}\mathbf{Q_{zz}^{-1}}\mathbf{Q_{zu}}$, $\boldsymbol{\Psi}_n = \mathbf{Y}(\mathbf{h_u}\mathbf{Y})^{-1}$ are terms that project the constraint into both spaces: range and nullspace. Note that $\mathbf{Q_z} = \mathbf{Z}^\top\mathbf{Q_u}$, $\mathbf{Q_{zx}} = \mathbf{Z}^\top\mathbf{Q_{ux}}$, $\mathbf{Q_{zu}} = \mathbf{Z}^\top\mathbf{Q_{uu}}$, $\mathbf{Q_{zz}} = \mathbf{Q_{zu}}\mathbf{Z}$ 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 $\mathbf{Q_{zz}^{-1}}$, $(\mathbf{h_u}\mathbf{Y})^{-1}$, and the constraint basis for the image and nullspace $[\mathbf{Y}\ \mathbf{Z}]$. 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, $\boldsymbol{\Psi}_n = \mathbf{Y}(\mathbf{h_u}\mathbf{Y})^{-1}$, $\boldsymbol{\Psi}_n\bar{\mathbf{h}}$, $\boldsymbol{\Psi}_n\mathbf{h_x}$, and $[\mathbf{Y}\ \mathbf{Z}]$ 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 $\mathcal{O}(N(n_x + n_u)^3)$, where n_x defines the dimension of the state and n_u of the control vector. Furthermore, these computations can be

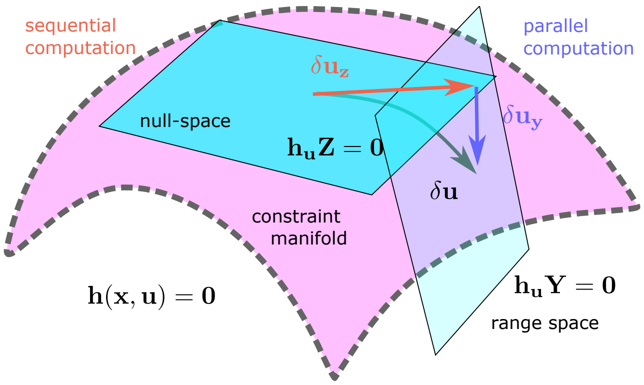

Fig. 4. Operations performed by our Riccati recursion based on a nullspace parametrization. The manifold of the inverse-dynamics constraint is nonconvex, smooth, and differentiable (pink geometry). When parametrizing the constraint using the fundamental basis of $\mathbf{h_u}$, we search for the direction along its null and range spaces (blue planes). This allows us to perform parallel computations for computing $\delta\mathbf{u_y}$ of each node, which reduces the algorithmic complexity of the Riccati recursion as we only need to compute $\delta\mathbf{u_z}$ in sequence. Note that, for the sake of clarity, we translate the *range-space* plane.

reused when the algorithm runs the Riccati recursion twice or more, e.g., when the Riccati recursion fails. Fig. 4 depicts the operations performed by a Riccati recursion based on our nullspace approach. Instead, Algorithm 1 (lines 3-5) provides further details on the quantities that can be computed in parallel with our nullspace factorization.

4) Benefits compared to a nullspace projection: Our approach uses a parametrization of the nullspace, in contrast to the *nullspace projection* proposed in [46]. It means that our approach does not need to pose a *singular optimal control* problem. It also performs more efficiently the Riccati recursion, as it does not require replacing the Cholesky decomposition with a Moore-Penrose pseudo-inverse, which is based on an expensive singular value decomposition.

D. Value function in a single node

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

$$\Delta\mathcal{V} = \Delta\mathcal{V}_1 + \frac{\Delta\mathcal{V}_2}{2} + \delta\mathbf{x}^\top\mathcal{V}_\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^\top\mathcal{V}_\mathbf{xx}\delta\mathbf{x} \tag{21}$$

with

$$\begin{aligned}
&\Delta\mathcal{V}_1 = -\boldsymbol{\pi}^\top\mathbf{Q_u}, \qquad\qquad \Delta\mathcal{V}_2 = \boldsymbol{\pi}^\top\mathbf{Q_{uu}}\boldsymbol{\pi},\\
&\mathcal{V}_\mathbf{x} = \mathbf{Q_x} + \boldsymbol{\Pi}^\top(\mathbf{Q_{uu}}\boldsymbol{\pi} - \mathbf{Q_u}) - \mathbf{Q}_\mathbf{ux}^\top\boldsymbol{\pi},\\
&\mathcal{V}_\mathbf{xx} = \mathbf{Q_{xx}} + (\boldsymbol{\Pi}^\top\mathbf{Q_{uu}} - 2\mathbf{Q}_\mathbf{ux}^\top)\boldsymbol{\Pi},
\end{aligned} \tag{22}$$

where $\boldsymbol{\pi}$, $\boldsymbol{\Pi}$ 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:

$$\begin{aligned}
\mathbf{x}_0^+ &= \tilde{\mathbf{x}}_0 \oplus (\alpha - 1)\bar{\mathbf{f}}_0,\\
\mathbf{u}_k^+ &= \mathbf{u}_k + \alpha\mathbf{k}_k + \mathbf{K}_k\delta\mathbf{x}_k^+,\\
\mathbf{x}_{k+1}^+ &= \mathbf{f}(\mathbf{x}_k^+, \mathbf{u}_k^+) \oplus (\alpha - 1)\bar{\mathbf{f}}_{k+1},
\end{aligned} \tag{23}$$

for $k = \{0, \cdots, N-1\}$ and $\delta\mathbf{x}_k^+ = \mathbf{x}_k^+ \ominus \mathbf{x}_k$, where a backtracking procedure tries different step lengths α; $\mathbf{x}_k^+$, $\mathbf{u}_k^+$ describes the potential new guess for the kth node; $\bar{\mathbf{f}}_0$ 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 evolution of the gaps in the system dynamics $\hat{\mathbf{f}}_s = (\bar{\mathbf{f}}_0, \cdots, \bar{\mathbf{f}}_N)$, 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 $\hat{\mathbf{f}}_s$, we compute a feasibility-aware expected improvement as

$$\Delta\ell(\alpha) = \alpha\sum_{k=0}^{N-1}\left(\Delta\ell_{1_k} + \frac{1}{2}\alpha\Delta\ell_{2_k}\right), \tag{24}$$

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

$$\begin{aligned}
\Delta\ell_{1_k} &= \boldsymbol{\pi}_k^\top\mathbf{Q}_{\mathbf{u}_k} + \bar{\mathbf{f}}_k^\top(\mathcal{V}_{\mathbf{x}_k} - \mathcal{V}_{\mathbf{xx}_k}\delta\mathbf{x}_k^+),\\
\Delta\ell_{2_k} &= \boldsymbol{\pi}_k^\top\mathbf{Q}_{\mathbf{uu}_k}\boldsymbol{\pi}_k + \bar{\mathbf{f}}_k^\top(2\mathcal{V}_{\mathbf{xx}_k}\delta\mathbf{x}_k^+ - \mathcal{V}_{\mathbf{xx}_k}\bar{\mathbf{f}}_k).
\end{aligned} \tag{25}$$

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

We then compute a merit function of the form:

$$\phi(\hat{\mathbf{x}}_s, \hat{\mathbf{u}}_s; \nu) = \sum_{k=0}^{N-1}\ell(\mathbf{x}_k, \mathbf{u}_k) + \nu\epsilon(\mathbf{x}_k, \mathbf{u}_k), \tag{26}$$

with

$$\epsilon(\mathbf{x}_k, \mathbf{u}_k) \coloneqq \|\bar{\mathbf{f}}_k\|_1 + \|\bar{\mathbf{h}}_k\|_1,$$

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

$$\nu \leftarrow \max\left(\nu, \frac{\Delta\ell(1)}{(1-\rho)\sum_{k=0}^{N-1}\epsilon(\mathbf{x}_k, \mathbf{u}_k)}\right), \tag{27}$$

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* sub-problem and $\sum_{k=0}^{N-1} \epsilon(\mathbf{x}_k, \mathbf{u}_k)$ as the reduction provided by the normal step.

Finally, we accept a step $\delta\hat{\mathbf{w}}_s^\alpha = \alpha[\delta\hat{\mathbf{x}}_s^\top \; \delta\hat{\mathbf{u}}_s^\top]^\top$ if the following Goldstein-inspired condition holds:

$$\phi(\hat{\mathbf{w}}_s^{+\alpha};\nu) - \phi(\hat{\mathbf{w}}_s;\nu) \leq \begin{cases} \eta_1 \Delta\Phi(\hat{\mathbf{w}}_s;\nu) & \text{if } \Delta\Phi(\hat{\mathbf{w}}_s;\nu) \leq 0 \\ \eta_2 \Delta\ell(\alpha) & \text{otherwise} \end{cases} \tag{28}$$

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

$$D(\phi(\hat{\mathbf{w}}_s;\nu); \delta\hat{\mathbf{w}}_s^\alpha) := \Delta\ell(\alpha) + \alpha \sum_{k=0}^{N-1} \epsilon(\mathbf{x}_k, \mathbf{u}_k) \tag{29}$$

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

F. Regularization and stopping criteria

To increase the algorithm's robustness, we regularize $\mathcal{V}'_{\mathbf{xx}}$ and $\mathcal{L}_{\mathbf{uu}}$ 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 μ, i.e.,

$$\mu \leftarrow \beta^{i,d}\mu, \quad \mathcal{V}'_{\mathbf{xx}} \leftarrow \mathcal{V}'_{\mathbf{xx}} + \mu\mathbf{f}_{\mathbf{x}}^\top\mathbf{f}_{\mathbf{x}}, \quad \mathcal{L}_{\mathbf{uu}} \leftarrow \mathcal{L}_{\mathbf{uu}} + \mu\mathbf{I},$$

where β^i and β^d are factors used to increase and decrease the regularization value μ, respectively. Our updating rule is as follows: we increase μ when the Cholesky decomposition in Eq. (20) (or in Eq. (17)) fails or when the forward pass accepts a step length smaller than α_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 μ when the forward pass accepts steps larger than α_1.

The $\mathcal{V}'_{\mathbf{xx}}$ term is defined as $\mathcal{L}_{\mathbf{xx}} + \mu\mathbf{f}_{\mathbf{x}}^\top\mathbf{f}_{\mathbf{x}}$, which can be also interpreted as a *banded regularization* as it corrects the matrix inertia that condenses the future nodes (i.e., $\mathcal{V}'_{\mathbf{xx}}$). 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:

$$\max\left(\sum_{k=0}^{N-1} \epsilon(\mathbf{x}_k, \mathbf{u}_k), |\Delta\ell(1)|\right). \tag{30}$$

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

Algorithm 1: Equality constrained DDP

/* derivatives and decompositions */
1 **for** $k \leftarrow 0$ **to** N **do in parallel**
2   residual and derivatives: $\bar{\mathbf{h}}$, $\bar{\mathbf{f}}$, $\ell_{\mathbf{p}}$, $\mathcal{L}_{\mathbf{pp}}$, $\mathbf{f}_{\mathbf{p}}$, $\mathbf{h}_{\mathbf{p}}$
3   **if** *nullspace factorization* **then**
4   nullspace and decomposition: $\mathbf{Y}$, $\mathbf{Z}$, $(\mathbf{h}_{\mathbf{u}}\mathbf{Y})^{-1}$
5   feed-forward/back in rank-space: $\mathbf{\Psi}_n\bar{\mathbf{h}}$, $\mathbf{\Psi}_n\mathbf{h}_{\mathbf{x}}$

/* compute search direction */
6 terminal value function: $\mathcal{V}'_{\mathbf{x}} \leftarrow \ell_{\mathbf{x}_N}$, $\mathcal{V}'_{\mathbf{xx}} \leftarrow \mathcal{L}_{\mathbf{xx}_N}$
7 **for** $k \leftarrow N-1$ **to** 0 **do**
8   local action-value function Eq. (13)
9   regularization: $\mathcal{V}'_{\mathbf{xx}}$, $\mathcal{L}_{\mathbf{uu}}$
10   **if** *nullspace factorization* **then**
11   nullspace action-value: $\mathbf{Q}_{\mathbf{z}}$, $\mathbf{Q}_{\mathbf{zz}}$, $\mathbf{Q}_{\mathbf{zx}}$, $\mathbf{Q}_{\mathbf{zu}}$
12   nullspace Cholesky: $\mathbf{Q}_{\mathbf{zz}}^{-1}$, $\mathbf{k}_n$, $\mathbf{K}_n$, $\tilde{\mathbf{Q}}_{\mathbf{zz}}$
13   feed-forward and feedback: $\boldsymbol{\pi}_n$, $\mathbf{\Pi}_n$ Eq. (20)
14   **else**
15   full-space Cholesky: $\mathbf{Q}_{\mathbf{uu}}^{-1}$, $\mathbf{k}$, $\mathbf{K}$
16   projected Cholesky: $(\mathbf{h}_{\mathbf{u}}\mathbf{Q}_{\mathbf{uu}}^{-1}\mathbf{h}_{\mathbf{u}}^\top)^{-1}$
17   free-space terms: $\mathbf{\Psi}_s$, $\mathbf{k}_s$, $\mathbf{K}_s$
18   feed-forward and feedback: $\boldsymbol{\pi}_s$, $\mathbf{\Pi}_s$ Eq. (17)
19   value function: $\Delta\mathcal{V}$, $\mathcal{V}_{\mathbf{x}}$, $\mathcal{V}_{\mathbf{xx}}$ Eq. (22)

/* try search direction */
20 **for** $\alpha \in \{1, \frac{1}{2}, \cdots, \frac{1}{2^n}\}$ **do**
21   **for** $k \leftarrow 0$ **to** N **do**
22   nonlinear rollout: $\mathbf{x}_k^+$, $\mathbf{u}_k^+$ Eq. (23)
23   expected improvement: $\Delta\ell(\alpha)$ Eq. (24)
24   merit penalty parameter: ν Eq. (27)
25   merit function: $\phi(\mathbf{x}_s, \mathbf{u}_s; \nu)$ Eq. (26)
26   **if** *success step* **then**
27   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 $\mathbf{h}_{\mathbf{u}}$, 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 $\mathbf{Y}$, $\mathbf{Z}$, $(\mathbf{h}_{\mathbf{u}}\mathbf{Y})^{-1}$, $\mathbf{\Psi}_n\bar{\mathbf{h}}$, and $\mathbf{\Psi}_n\mathbf{h}_{\mathbf{x}}$ (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 $\mathbf{q}$, $\mathbf{v}$, $\mathbf{a}$, $\boldsymbol{\tau}$, and $\boldsymbol{\lambda}$. The first couple of variables naturally describe the state of the system $\mathbf{x} = [\mathbf{q}^\top \ \mathbf{v}^\top]^\top$, while the latter ones refer to its control inputs $\mathbf{u} = [\mathbf{a}^\top \ \boldsymbol{\tau}^\top \ \boldsymbol{\lambda}^\top]^\top$. Therefore, the linearization of the nonlinear equality constraints has the form:

$$\overbrace{\begin{bmatrix} \frac{\partial \mathrm{IDA}}{\partial \mathbf{q}} & \frac{\partial \mathrm{IDA}}{\partial \mathbf{v}} \\ -\frac{\partial \mathbf{a}_c}{\partial \mathbf{q}} & -\frac{\partial \mathbf{a}_c}{\partial \mathbf{v}} \end{bmatrix}}^{\mathbf{h}_\mathbf{x}} \begin{bmatrix} \delta\mathbf{q} \\ \delta\mathbf{v} \end{bmatrix} + \overbrace{\begin{bmatrix} \mathbf{M} & -\frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}} & -\mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} & \mathbf{0} \end{bmatrix}}^{\mathbf{h}_\mathbf{u}^{\text{redundant}}} \begin{bmatrix} \delta\mathbf{a} \\ \delta\boldsymbol{\tau} \\ \delta\boldsymbol{\lambda} \end{bmatrix} = -\begin{bmatrix} \bar{\mathbf{h}}_{\mathrm{ID}} \\ \bar{\mathbf{h}}_{\boldsymbol{\lambda}} \end{bmatrix}, \tag{31}$$

with

$$\frac{\partial \mathrm{IDA}}{\partial \mathbf{q}} = \frac{\partial \mathrm{ID}}{\partial \mathbf{q}} - \frac{\partial \mathrm{A}}{\partial \mathbf{q}}, \qquad \frac{\partial \mathrm{IDA}}{\partial \mathbf{v}} = \frac{\partial \mathrm{ID}}{\partial \mathbf{v}} - \frac{\partial \mathrm{A}}{\partial \mathbf{v}},$$

where $\bar{\mathbf{h}}_{\mathrm{ID}}$, $\bar{\mathbf{h}}_{\boldsymbol{\lambda}}$ are the values of joint efforts and contact acceleration at the linearization point; $\frac{\partial \mathrm{ID}}{\partial \mathbf{q}}$, $\frac{\partial \mathrm{ID}}{\partial \mathbf{v}}$ are the RNEA derivatives [9]; $\frac{\partial \mathbf{a}_c}{\partial \mathbf{q}}$, $\frac{\partial \mathbf{a}_c}{\partial \mathbf{v}}$ are the derivatives of the frame acceleration; and $\frac{\partial \mathrm{A}}{\partial \mathbf{q}}$, $\frac{\partial \mathrm{A}}{\partial \mathbf{v}}$, $\frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}$ are the derivatives of the arbitrary actuation model $\mathrm{A}(\mathbf{q}, \mathbf{v}, \boldsymbol{\tau})$. The dimension of the control input is $n_v + n_j + n_f$ with n_v 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. $n_x + n_u$). We can do so by assuming that the RNEA constraint is always feasible (i.e., $\bar{\mathbf{h}}_{\mathrm{ID}} = \mathbf{0}$), which leads to the following expression

$$\delta\boldsymbol{\tau} = \left(\frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}\right)^\dagger \left(\frac{\partial \mathrm{IDA}}{\partial \mathbf{q}}\delta\mathbf{q} + \frac{\partial \mathrm{IDA}}{\partial \mathbf{v}}\delta\mathbf{v}\right), \tag{32}$$

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

$$\overbrace{\begin{bmatrix} \mathbf{S}\frac{\partial \mathrm{IDA}}{\partial \mathbf{q}} & \mathbf{S}\frac{\partial \mathrm{IDA}}{\partial \mathbf{v}} \\ -\frac{\partial \mathbf{a}_c}{\partial \mathbf{q}} & -\frac{\partial \mathbf{a}_c}{\partial \mathbf{v}} \end{bmatrix}}^{\mathbf{h}_\mathbf{x}} \begin{bmatrix} \delta\mathbf{q} \\ \delta\mathbf{v} \end{bmatrix} + \overbrace{\begin{bmatrix} \mathbf{M} & -\mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix}}^{\mathbf{h}_\mathbf{u}^{\text{condensed}}} \begin{bmatrix} \delta\mathbf{a} \\ \delta\boldsymbol{\lambda} \end{bmatrix} = -\begin{bmatrix} \bar{\mathbf{h}}_{\mathrm{A}} \\ \bar{\mathbf{h}}_{\boldsymbol{\lambda}} \end{bmatrix}, \tag{33}$$

with $\bar{\mathbf{h}}_{\mathrm{A}}$ as the value of the under-actuated efforts at the linearization point and $\mathbf{S} = \mathbf{I} - \frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}\left(\frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}\right)^\dagger$ 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, $\frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}$ 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 $\mathbf{u} = [\mathbf{a}^\top \ \boldsymbol{\lambda}^\top]^\top$. 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.,

$$\mathbf{f}_\mathbf{u} = \begin{bmatrix} \mathbf{f}_\mathbf{a} & \mathbf{0} \end{bmatrix}, \tag{34}$$

where $\mathbf{f}_\mathbf{a}$ is the partial derivatives with respect to the generalized accelerations, and the null block represents the partial derivatives of the remaining components (i.e., $\boldsymbol{\tau}$ and/or $\boldsymbol{\lambda}$). Furthermore, the structure of $\mathbf{f}_\mathbf{a}$ 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

$$\mathbf{f}_\mathbf{u}^\top \mathcal{V}_\mathbf{x}^+ = \begin{bmatrix} \mathbf{f}_\mathbf{a}^\top \mathcal{V}_\mathbf{x}^+ \\ \mathbf{0} \end{bmatrix}, \quad \mathbf{f}_\mathbf{u}^\top \mathcal{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} = \begin{bmatrix} \mathbf{f}_\mathbf{a}^\top \mathcal{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{x} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix},$$
$$\mathbf{f}_\mathbf{u}^\top \mathcal{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{u} = \begin{bmatrix} \mathbf{f}_\mathbf{a}^\top \mathcal{V}'_{\mathbf{xx}} \mathbf{f}_\mathbf{a} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{35}$$

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 $400\,\mathrm{Hz}$. 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 placements given the terrain map. We use tools from mixed-integer convex (MIC) and quadratic programming to plan a sequence of footsteps in real time. This approach is inspired by our previous work [69]. Concretely, our system plans six footstep placements and regions at $10\,\mathrm{Hz}$ and the quadratic program adapts the swing-foot trajectories within the MPC horizon at $50\,\mathrm{Hz}$.

3) Inverse-dynamics MPC and feedback policy: The inverse-dynamics MPC (our contribution) computes whole-body motions, contact forces, and feedback policies for generalized acceleration and contact forces at a fixed optimization horizon ($1\,\mathrm{s}$ in our experiments). The *feedback-policy* controller builds a control policy in the *joint-effort space*. In order to do so, we map the MPC policy into the joint-effort space based on generalized acceleration and contact forces. This equivalent policy is a function composition of the MPC policy and the *inverse-dynamics* function, i.e.,

$$\boldsymbol{\pi}^*_{\boldsymbol{\tau}}(\mathbf{x}) = \mathrm{ID}^* \circ \boldsymbol{\pi}^*_{\mathrm{MPC}}(\mathbf{x}), \tag{36}$$

where $\circ$ is the composition operator, ID^* is the inverse dynamics at the optimal solution $(\mathbf{x}^*, \mathbf{u}^*)$, $\boldsymbol{\pi}^*_{\mathrm{MPC}}$ is the policy computed by the MPC, and $\boldsymbol{\pi}^*_{\boldsymbol{\tau}}$ is the same policy expressed in the joint-effort space. Note that $\boldsymbol{\pi}^*_{\boldsymbol{\tau}}$ and $\boldsymbol{\pi}^*_{\mathrm{MPC}}$ contains the feed-forward and feedback terms computed using our nullspace factorization for equality-constrained DDP. Below we provide more details about our inverse-dynamics MPC and feedback policy.

B. MPC formulation

At each MPC step, we solve an OC problem with inverse dynamics. It computes whole-body motions and contact forces given a predefined footstep plan as follows:

$$\begin{aligned}
\min_{\hat{\mathbf{x}}_s, \hat{\mathbf{u}}_s} \; & \sum_{k=0}^{N-1} \|\mathbf{q}_k \ominus \mathbf{q}_{\text{nom}}\|^2_{\mathbf{Q}} + \|\mathbf{v}_k\|^2_{\mathbf{N}} + \|\mathbf{u}_k\|^2_{\mathbf{R}} + \|\boldsymbol{\lambda}_{\mathcal{C}_k}\|^2_{\mathbf{K}} \\
\text{s.t. } & \mathbf{q}_0 = \tilde{\mathbf{q}}_0, && \text{(initial pos.)} \\
& \mathbf{v}_0 = \tilde{\mathbf{v}}_0, && \text{(initial vel.)} \\
& \text{if contact-gain transition:} \\
& \quad \mathbf{q}_{k+1} = \mathbf{q}_k, \\
& \quad \begin{bmatrix} \mathbf{v}_{k+1} \\ -\boldsymbol{\lambda}_{\mathcal{C}_k} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_k & \mathbf{J}^\top_{\mathcal{C}_k} \\ \mathbf{J}_{\mathcal{C}_k} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}^{\mathcal{I}}_{bias_k} \\ -\mathbf{a}^{\mathcal{I}}_{\mathcal{C}_k} \end{bmatrix}, && \text{(impulse dyn.)} \\
& \text{else:} \\
& \quad [\mathbf{q}_{k+1}, \mathbf{v}_{k+1}] = \boldsymbol{\psi}(\mathbf{q}_k, \mathbf{v}_k, \mathbf{a}_k), && \text{(kin. integrator)} \\
& \quad \mathrm{ID}(\mathbf{q}_k, \mathbf{v}_k, \mathbf{a}_k, \boldsymbol{\lambda}_{\mathcal{C}_k}) = \mathbf{0}, && \text{(inverse dyn.)} \\
& \quad \boldsymbol{\lambda}_{\mathcal{G}_k} = \mathbf{0}, && \text{(non-contact for.)} \\
& \quad \mathbf{C}\boldsymbol{\lambda}_{\mathcal{C}_k} \geq \mathbf{c}, && \text{(friction-cone)} \\
& \quad \log({}^{\mathcal{W}}\mathbf{p}^{-1}_{\mathcal{C},\mathcal{G}_k} \cdot {}^{\mathcal{W}}\mathbf{p}^{ref}_{\mathcal{C},\mathcal{G}_k}) = \mathbf{0}, && \text{(contact pos.)} \\
& \quad {}^{\mathcal{W}}\dot{\mathbf{p}}^{-1}_{\mathcal{C},\mathcal{G}_k} - {}^{\mathcal{W}}\dot{\mathbf{p}}^{ref}_{\mathcal{C},\mathcal{G}_k} = \mathbf{0}, && \text{(contact velocity)} \\
& \quad \underline{\mathbf{x}} \leq \mathbf{x}_k \leq \bar{\mathbf{x}}, && \text{(state bounds)} \\
& \quad \underline{\mathbf{u}} \leq \mathbf{u}_k \leq \bar{\mathbf{u}}, && \text{(control bounds)}
\end{aligned} \tag{37}$$

where $\mathbf{q}_{\text{nom}}$ defines the robot's nominal posture, the control inputs are defined by generalized acceleration and contact forces $\mathbf{u} = [\mathbf{a}^\top \; \boldsymbol{\lambda}^\top]^\top$ (i.e., condensed inverse dynamics), $(\hat{\mathbf{x}}_s, \hat{\mathbf{u}}_s)$ describes the state trajectory and control sequence, N is the optimization horizon, the linearized friction cone is defined by $(\mathbf{C}, \mathbf{c})$, $\log(\cdot)$ operator defines the logarithmic map needed to handle contact placement that lies on a $\mathbb{SE}(3)$ manifold, ${}^{\mathcal{W}}\mathbf{p}^{-1}_{\mathcal{C},\mathcal{G}} \cdot {}^{\mathcal{W}}\mathbf{p}^{ref}_{\mathcal{C},\mathcal{G}}$ describes the inverse composition between the reference and current contact placements [70], $(\underline{\mathbf{x}}, \bar{\mathbf{x}})$ are the state bounds, $(\underline{\mathbf{u}}, \bar{\mathbf{u}})$ are the control bounds, $(\mathbf{Q}, \mathbf{N}, \mathbf{R}, \mathbf{K})$ are weighing matrices used in the different cost terms, $\mathcal{C}_k$ and $\mathcal{G}_k$ are the set of active and swing contacts given the kth node. Both contact positions and velocities $\left({}^{\mathcal{W}}\mathbf{p}_{\mathcal{C},\mathcal{G}}, {}^{\mathcal{W}}\dot{\mathbf{p}}_{\mathcal{C},\mathcal{G}}\right)$ are expressed in the inertial frame $\mathcal{W}$.

1) Inverse dynamics and contact forces: Eq. (37) formulates inverse dynamics as a function of the *kinematic integrator* $\boldsymbol{\psi}$ 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

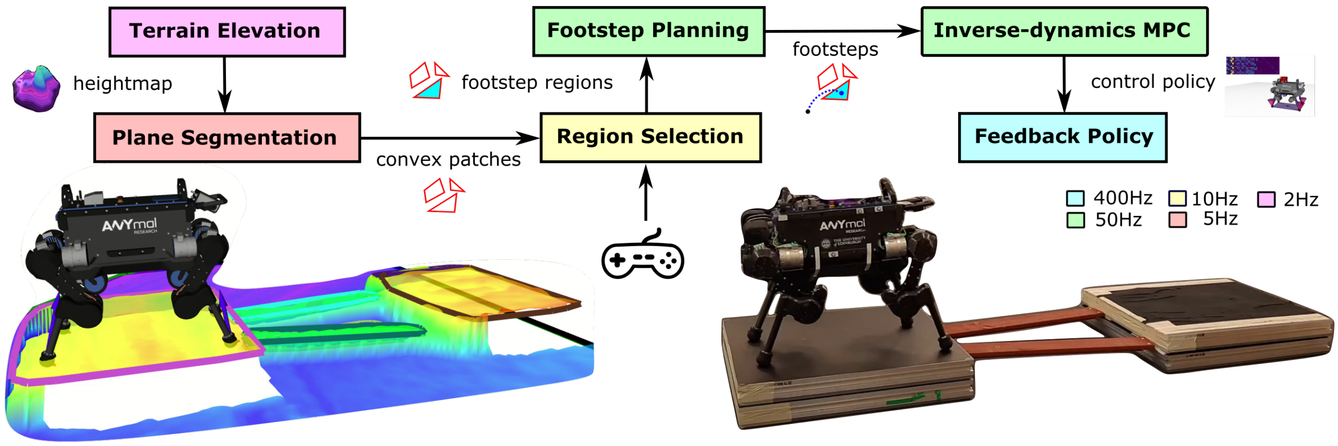

Fig. 5. Overview of our *perceptive locomotion* pipeline. Our inverse-dynamics MPC computes updated policies, maps them to joint space, and transmits them to a *feedback-policy* controller that closes the loop with state estimation and sensing. The footstep planner computes swing-foot trajectories that reach a selected *footstep region* and avoid obstacles. These selected regions are chosen to track the desired robot velocity (commanded from a joystick) while guaranteeing footstep feasibility. We use different colors to describe the frequency of each module in our locomotion pipeline. Our key contribution is an inverse-dynamics MPC (Section VI-B) that combines a novel nullspace factorization for equality-constrained DDP (Section IV-B), condensed inverse dynamics (Section V-A), and a feedback policy in the joint-effort space (Section VI-C). More details of our perceptive locomotion pipeline will be presented in an upcoming publication.

results in practice. These barrier functions are defined as

$$\Phi_{\text{cone}}(\mathbf{x},\mathbf{u}) = w_{\text{cone}}\sum_{i=0}^{m_{\text{cone}}}\begin{cases}\{\mathbf{C}\boldsymbol{\lambda}_{\mathcal{C}}-\mathbf{c}\}_i^2 & \text{if } \{\mathbf{C}\boldsymbol{\lambda}_{\mathcal{C}}-\mathbf{c}\}_i \leq 0\\ 0 & \text{otherwise}\end{cases},$$

$$\Phi_{\text{state}}(\mathbf{x}) = w_{\text{state}}\sum_{i=0}^{n_x}\begin{cases}\{\mathbf{x}-\bar{\mathbf{x}}\}_i^2 & \text{if } \{\mathbf{x}-\bar{\mathbf{x}}\}_i \leq 0\\ \{\underline{\mathbf{x}}-\mathbf{x}\}_i^2 & \text{if } \{\underline{\mathbf{x}}-\mathbf{x}\}_i \leq 0\\ 0 & \text{otherwise}\end{cases},$$

$$\Phi_{\text{control}}(\mathbf{u}) = w_{\text{control}}\sum_{i=0}^{n_u}\begin{cases}\{\mathbf{u}-\bar{\mathbf{u}}\}_i^2 & \text{if } \{\mathbf{u}-\bar{\mathbf{u}}\}_i \leq 0\\ \{\underline{\mathbf{u}}-\mathbf{u}\}_i^2 & \text{if } \{\underline{\mathbf{u}}-\mathbf{u}\}_i \leq 0\\ 0 & \text{otherwise}\end{cases}, \tag{38}$$

where $\{\cdot\}_i$ is the value of the ith element; m_{cone} is the number of facets of the friction cone; w_{cone}, w_{state} and w_{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: $w_{\text{cpos}}\|{}^{\mathcal{W}}\mathbf{p}_{\mathcal{G}}^{ref} \ominus {}^{\mathcal{W}}\mathbf{p}_{\mathcal{G}}\|^2$ and $w_{\text{cvel}}\|{}^{\mathcal{W}}\dot{\mathbf{p}}_{\mathcal{C},\mathcal{G}_k}^{-1} - {}^{\mathcal{W}}\dot{\mathbf{p}}_{\mathcal{C},\mathcal{G}_k}^{ref}\|$ with w_{cpos} and w_{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 $1\,\text{s}$, which is described through 100 nodes with timesteps of $10\,\text{ms}$. 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 $1\,\text{ms}$), 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 μ. This is to encourage a similar numerical evolution to that seen in the optimal control case. Instead, the penalty parameter ν 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.,

$$\begin{bmatrix}\delta\mathbf{a}^*\\ \delta\boldsymbol{\lambda}^*\end{bmatrix} = -\begin{bmatrix}\boldsymbol{\pi}_{\mathbf{a}}\\ \boldsymbol{\pi}_{\boldsymbol{\lambda}}\end{bmatrix} - \begin{bmatrix}\boldsymbol{\Pi}_{\mathbf{a}}\\ \boldsymbol{\Pi}_{\boldsymbol{\lambda}}\end{bmatrix}\delta\mathbf{x}, \tag{39}$$

where $\boldsymbol{\pi}_{\mathbf{a}}$, $\boldsymbol{\pi}_{\boldsymbol{\lambda}}$ are the feed-forward commands for the reference generalized acceleration $\mathbf{a}^*$ and contact forces $\boldsymbol{\lambda}^*$, respectively, and $\boldsymbol{\Pi}_{\mathbf{a}}$, $\boldsymbol{\Pi}_{\boldsymbol{\lambda}}$ 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.,

$$\begin{aligned}\delta\boldsymbol{\tau}^* &= -\boldsymbol{\pi}_{\boldsymbol{\tau}} - \boldsymbol{\Pi}_{\boldsymbol{\tau}}\delta\mathbf{x},\\ &= (\mathrm{A}_{\boldsymbol{\tau}}^*)^\dagger \cdot \mathrm{ID}(\mathbf{q}^*,\mathbf{v}^*,\delta\mathbf{a}^*,\delta\boldsymbol{\lambda}^*),\end{aligned} \tag{40}$$

with $\mathrm{A}_{\boldsymbol{\tau}}^* = \frac{\partial \mathrm{A}}{\partial \boldsymbol{\tau}}\Big|_{\mathbf{q}^*,\mathbf{v}^*}$, is the selection matrix linearized around the optimal state that is described by $\mathbf{q}^*$, $\mathbf{v}^*$. This boils down to two procedures, one for each term: feed-forward and feedback. First, we compute the feed-forward joint effort commands $\boldsymbol{\pi}_{\boldsymbol{\tau}}$ by injecting only the feed-forward terms, i.e.,

$$\begin{aligned}\boldsymbol{\pi}_{\boldsymbol{\tau}} &= (\mathrm{A}^*_{\boldsymbol{\tau}})^{\dagger} \cdot \mathrm{ID}(\mathbf{q}^*, \mathbf{v}^*, \boldsymbol{\pi}_{\mathbf{a}}, \boldsymbol{\pi}_{\boldsymbol{\lambda}}), \\ &= (\mathrm{A}^*_{\boldsymbol{\tau}})^{\dagger} \cdot \left(\mathbf{M}(\mathbf{q}^*)\boldsymbol{\pi}_{\mathbf{a}} + \mathbf{h}(\mathbf{q}^*, \mathbf{v}^*) - \mathbf{J}_c^{\top}(\mathbf{q}^*)\boldsymbol{\pi}_{\boldsymbol{\lambda}}\right), \quad (41)\end{aligned}$$

where $\mathbf{h}(\mathbf{q}^*, \mathbf{v}^*)$ describes the Coriolis and gravitational forces, $\mathbf{M}(\mathbf{q}^*)$ the joint-space inertia matrix, and $\mathbf{J}_c(\mathbf{q}^*)$ the stack of contact Jacobians evaluated in the optimal robot configuration $\mathbf{q}^*$ and velocity $\mathbf{v}^*$. Second, we calculate the feedback joint torque gains $\boldsymbol{\Pi}_{\boldsymbol{\tau}}$ using a similar procedure, i.e., by injecting the feedback terms as follows:

$$\boldsymbol{\Pi}_{\boldsymbol{\tau}} = (\mathrm{A}^*_{\boldsymbol{\tau}})^{\dagger} \cdot \left(\mathbf{M}(\mathbf{q}^*)\boldsymbol{\Pi}_{\mathbf{a}} - \mathbf{J}_c^{\top}(\mathbf{q}^*)\boldsymbol{\Pi}_{\boldsymbol{\lambda}}\right). \quad (42)$$

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.

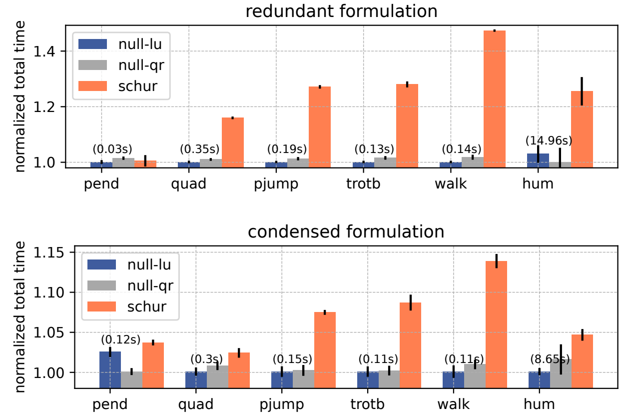

Fig. 6. Normalized computation time for solving different optimal control problems with different factorization approaches. For all cases, we observe that the nullspace factorizations (`null-lu/qr`) are always faster than the Schur-complement one (`schur`). Furthermore, there is a higher reduction in computation time with redundant formulations as the number of constraints increases. Indeed, the computation reduction for the quadrupedal walking problem (`walk`) is 47.3%, which is higher than the pace-jump (27%, `pjump`) or trot-bound (27.9%, `trotb`) cases. Finally, we include the minimum computation time for each problem at the top of the bar charts.

VII. Results

We implemented in C++ both factorizations for the equality-constrained DDP algorithm as well as both inverse-dynamics formulations within the CROCODDYL 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 × 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: $\rho = 0.3$, $\eta_1 = 0.1$, $\eta_2 = 2$, $\alpha_0 = 0.01$, $\alpha_1 = 0.5$, $\kappa_0 = 10^{-4}$, $\beta^i = 10^6$ 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 and its standard deviation over 500 trials using 8 threads and the same initialization. We expect a higher improvement if we increase the number of threads. We provided results for nullspace factorizations based on both rank-revealing decompositions: LU with full pivoting (`null-lu`)

TABLE I. Percentage of successful resolutions, average cost and iterations using random initialization when the solver handles or not the dynamics feasibility.

	with dyn. feasibility			without dyn. feasibility		
Formulations	Iter.	Cost	Succ.	Iter.	Cost	Succ.
redundant	**11.8**	**2838**	**100%**	27.5	3031	91.8%
condensed	**11.9**	**2839**	**100%**	28.4	3069	91%

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 $(\mathbf{h_u}\mathbf{Y})^{-1}$ efficiently, as this matrix is square-invertible. If not indicated otherwise, we use LU with full pivoting decomposition to compute $[\mathbf{Y}, \mathbf{Z}]$.

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 ℓ_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 CROCODDYL 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 II. Number of iterations required by different problem discretizations and formulations. The most robust formulation is highlighted in bold.

		frequencies (Hz)						
Problems		100	80	60	50	40	20	10
`quad`	forward	10	10	11	11	12	14	17
	inverse	6	6	6	6	6	8	9
`pjump`	forward	20	21	23	24	40	✗	✗
	inverse	20	19	21	22	19	41	199
`trotb`	forward	15	16	18	25	99	✗	✗
	inverse	9	9	12	16	36	153	199
`walk`	forward	10	9	10	10	12	48	99
	inverse	9	10	12	8	50	14	92

✗ algorithm does not find a solution within 200 iterations.

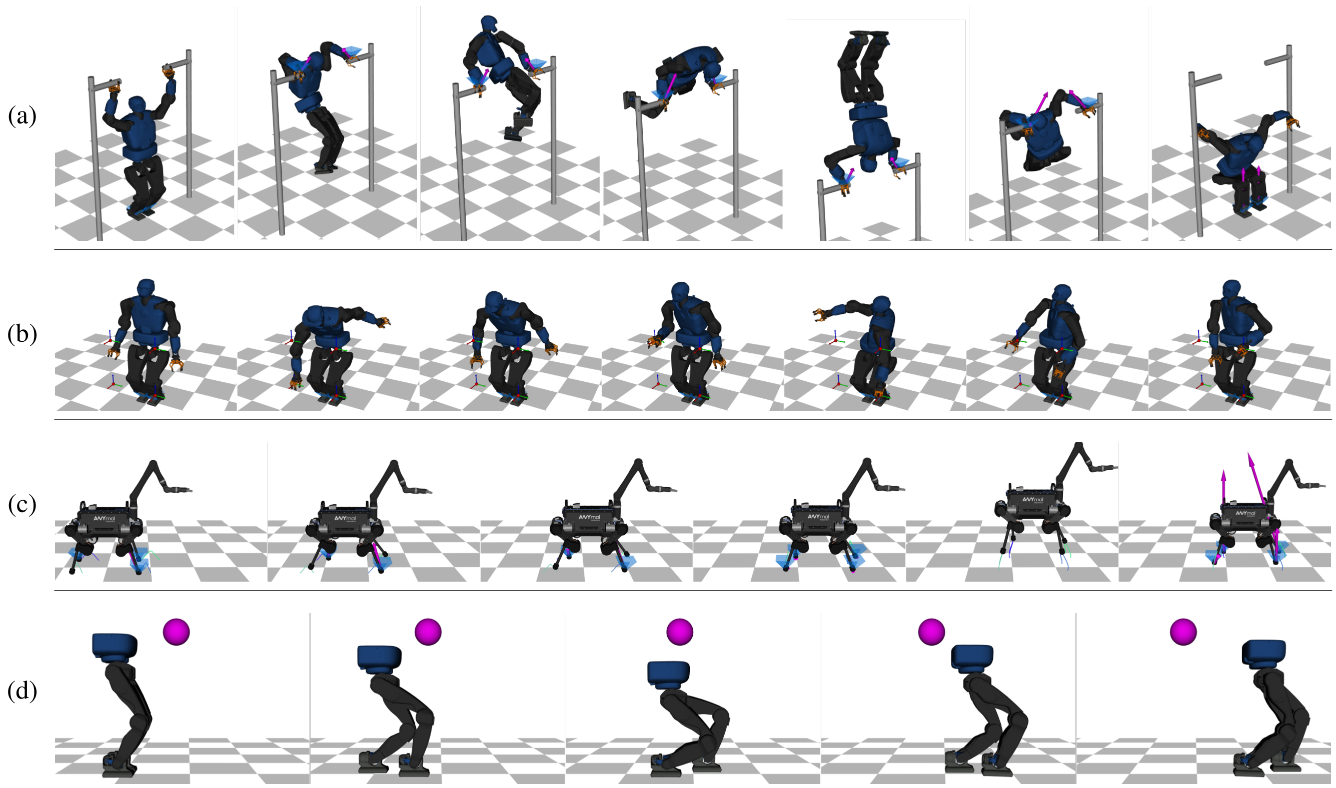

Fig. 7. Snapshots of different maneuvers computed using our inverse-dynamics formulations and equality-constrained DDP. (a) A gymnastic routine performed by the Talos robot. (b) Reaching a few grasping points with the Talos robot. (c) A sequence of walking, trotting and jumping gaits executed by the ANYmal robot. (d) Talos legs squatting to avoid an obstacle. To watch the video, click the picture or see `https://youtu.be/NhvSUVopPCI`.

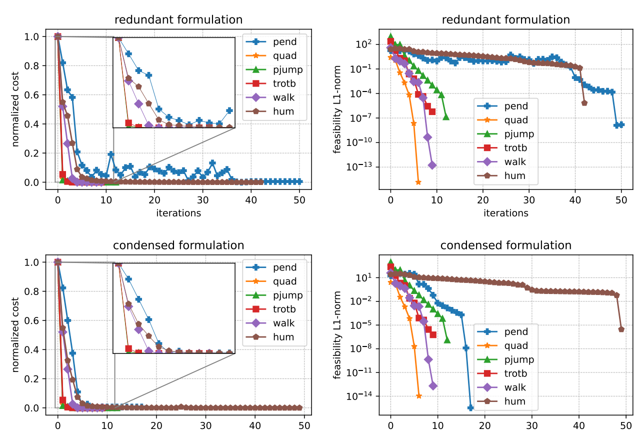

Fig. 8. Cost and feasibility evolution for redundant and condensed formulations. (Left) Normalized cost per iteration. (Right) ℓ_1-norm of the total feasibility (both dynamics and equality constraints) per iteration. A redundant formulation allows our solver to handle the entire problem's feasibility and optimality, while a condensed one imposes RNEA feasibility. This is key to reducing the total cost of the `pend` and `hum` problems.

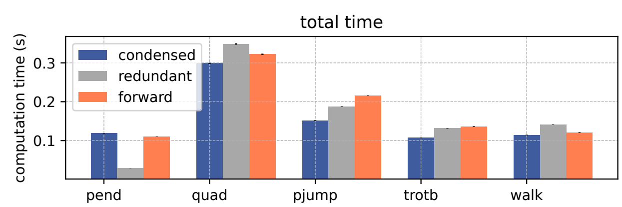

Fig. 9. Total computation time for solving different optimal control problems with different formulations. Except for `pend`, the condensed inverse-dynamics formulation solved these problems faster than the forward-dynamics formulations.

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

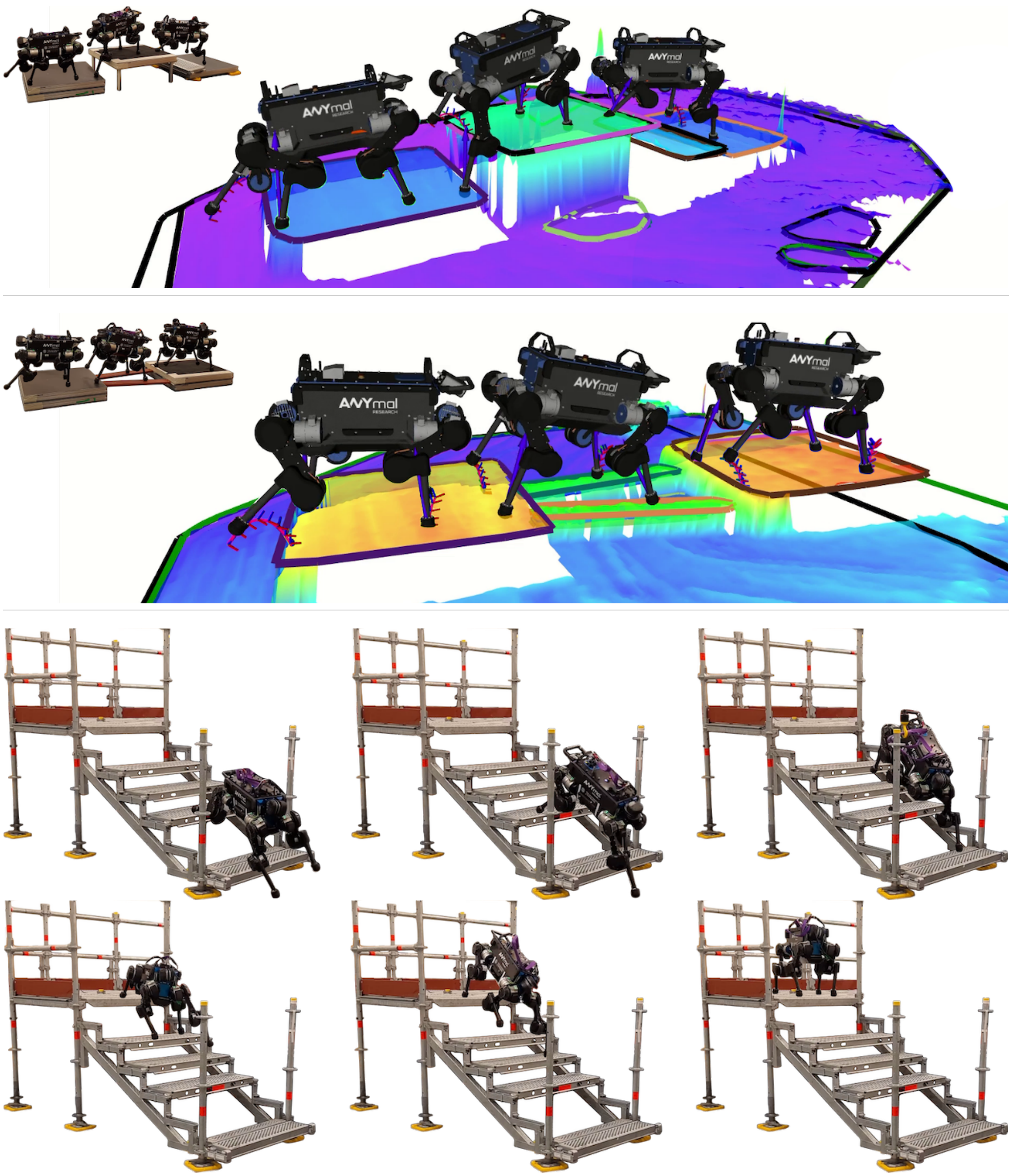

Fig. 10. Snapshots of different locomotion maneuvers computed by our inverse-dynamics MPC. All these experimental trials use the feedback-policy controller and onboard state estimation and perception. (a) Walking over a set of pallets of different heights. (b) Crossing a gap between two beams. (c) Climbing up industrial stairs with missing steps. To watch the video, click the picture or see `https://youtu.be/NhvSUVopPCI?t=161`.

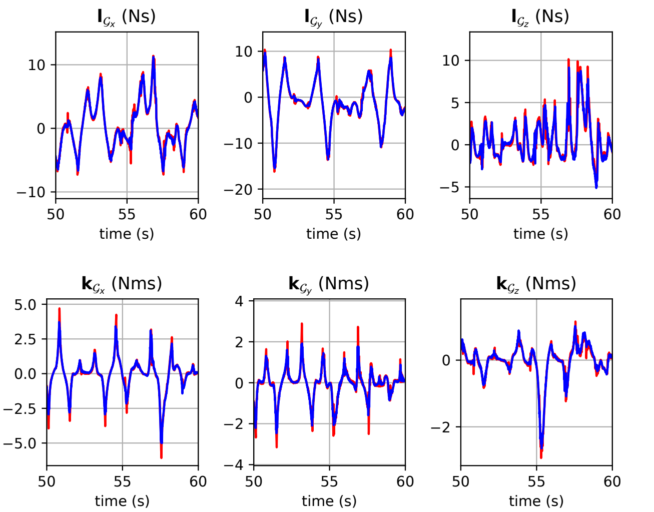

Fig. 11. Linear (top) and angular (bottom) momentum tracking of our feedback-policy controller when the ANYmal robot crossed the first missed tread in the stair-climbing experiment. Reference/desired signals are blue, while measured signals are red. Our feedback-policy controller accurately tracks the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

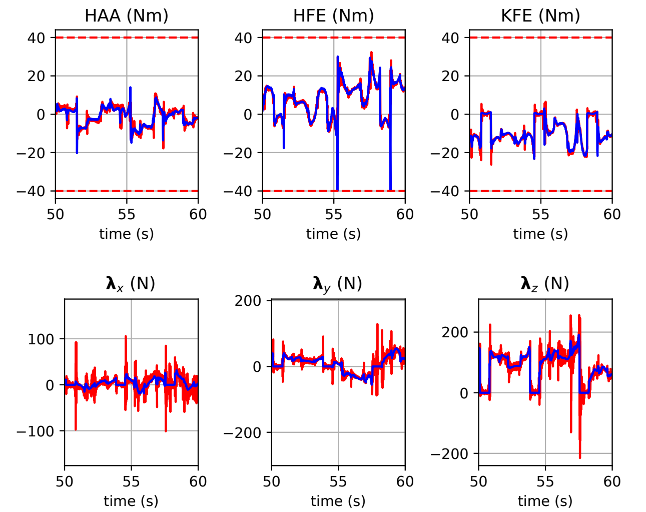

Fig. 12. Joint torque and force tracking of the ANYmal's right-front leg when the robot crossed the first missed tread in the stair-climbing experiment. Reference/desired signals are blue, while measured signals are red. Our feedback-policy controller accurately tracks joint torque commands and desired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

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,

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).

joint efforts, and swing-foot motions. Fig. 11 shows linear and angular momentum tracking when the ANYmal robot crossed the first missed tread in Fig. 10c. Instead, Fig. 12 shows torque and RF-foot force tracking during the same moment.

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

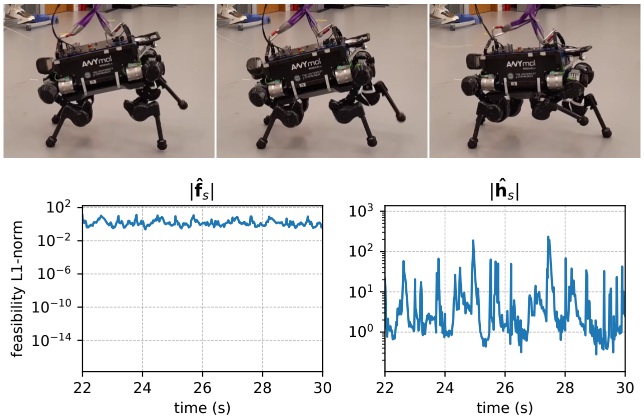

Fig. 14. Kinematics and dynamics feasibility evolution during multiple body disturbances while the ANYmal was walking. Every time that we pushed the robot, our inverse-dynamics MPC computed the optimal trajectories and feedback policies to balance the ANYmal robot while walking dynamically in place (top). When this happens, our solver increases the infeasibility instantly, which helps to quickly find a solution (bottom). The bottom plots show the ℓ_1-norm of the kinematics $|\bar{\mathbf{f}}_s|$ and dynamics $|\bar{\mathbf{h}}_s|$ 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 $10\,\mathrm{Hz}$ of trajectory discretization). Despite that computing the image and kernel $[\mathbf{Y}, \mathbf{Z}]$ 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.

[1]The backward pass is the routine that computes the search direction (see Algorithm 1).

Appendix A

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

$$\boldsymbol{\xi}^+_k = \mathcal{V}_{\mathbf{x}_{k+1}} + \mathcal{V}_{\mathbf{xx}_{k+1}}\delta\mathbf{x}_{k+1} \quad \forall k = \{N, N-1, \cdots, 1\}.$$

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

$$\begin{bmatrix} -\mathbf{I} & \mathcal{L}_{\mathbf{xx}_N} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^+_{N-1} \\ \delta\mathbf{x}_N \end{bmatrix} = -\boldsymbol{\ell}_{\mathbf{x}_N}.$$

By definition, we have $\mathcal{V}_{\mathbf{x}_N} = \boldsymbol{\ell}_{\mathbf{x}_N}$ and $\mathcal{V}_{\mathbf{xx}_N} = \mathcal{L}_{\mathbf{xx}_N}$, which both represent the relationship above at the terminal node. Without sacrificing generality, we compute the remaining derivatives of the value function $\mathcal{V}_{\mathbf{x}_k}$ and $\mathcal{V}_{\mathbf{xx}_k}$ recursively by solving the next sub-problem, i.e.,

$$\begin{bmatrix} -\mathbf{I} & \mathcal{L}_{\mathbf{xx}_k} & \mathcal{L}^\top_{\mathbf{ux}_k} & \mathbf{f}^\top_{\mathbf{x}_k} & \\ & \mathcal{L}_{\mathbf{ux}_k} & \mathcal{L}_{\mathbf{uu}_k} & \mathbf{f}^\top_{\mathbf{u}_k} & \\ & \mathbf{f}_{\mathbf{x}_k} & \mathbf{f}_{\mathbf{u}_k} & & -\mathbf{I} \\ & & & -\mathbf{I} & \mathcal{V}_{\mathbf{xx}_{k+1}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^+_{k-1} \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \\ \boldsymbol{\xi}^+_k \\ \delta\mathbf{x}_{k+1} \end{bmatrix} = -\begin{bmatrix} \boldsymbol{\ell}_{\mathbf{x}_k} \\ \boldsymbol{\ell}_{\mathbf{u}_k} \\ \bar{\mathbf{f}}_k \\ \mathcal{V}_{\mathbf{x}_{k+1}} \end{bmatrix}$$

Then, condensing the third and fourth rows yields

$$\begin{bmatrix} -\mathbf{I} & \mathbf{Q}_{\mathbf{xx}_k} & \mathbf{Q}^\top_{\mathbf{ux}_k} \\ & \mathbf{Q}_{\mathbf{ux}_k} & \mathbf{Q}_{\mathbf{uu}_k} \end{bmatrix} \begin{bmatrix} \boldsymbol{\xi}^+_{k-1} \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix} = -\begin{bmatrix} \mathbf{Q}_{\mathbf{x}_k} \\ \mathbf{Q}_{\mathbf{u}_k} \end{bmatrix}.$$

with

$$\begin{aligned} \mathbf{Q}_{\mathbf{x}_k} &= \boldsymbol{\ell}_{\mathbf{x}_k} + \mathbf{f}^\top_{\mathbf{x}_k}(\mathcal{V}_{\mathbf{x}_{k+1}} + \mathcal{V}_{\mathbf{xx}_{k+1}}\bar{\mathbf{f}}_k), \\ \mathbf{Q}_{\mathbf{u}_k} &= \boldsymbol{\ell}_{\mathbf{u}_k} + \mathbf{f}^\top_{\mathbf{u}_k}(\mathcal{V}_{\mathbf{x}_{k+1}} + \mathcal{V}_{\mathbf{xx}_{k+1}}\bar{\mathbf{f}}_k), \\ \mathbf{Q}_{\mathbf{xx}_k} &= \mathcal{L}_{\mathbf{xx}_k} + \mathbf{f}^\top_{\mathbf{x}_k}\mathcal{V}_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{x}_k}, \\ \mathbf{Q}_{\mathbf{ux}_k} &= \mathcal{L}_{\mathbf{ux}_k} + \mathbf{f}^\top_{\mathbf{u}_k}\mathcal{V}_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{x}_k}, \\ \mathbf{Q}_{\mathbf{uu}_k} &= \mathcal{L}_{\mathbf{uu}_k} + \mathbf{f}^\top_{\mathbf{u}_k}\mathcal{V}_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{u}_k}. \end{aligned}$$

This can be further condensed by injecting the optimal policy

$$\delta\mathbf{u}_k = -\boldsymbol{\pi}_k - \boldsymbol{\Pi}_k\delta\mathbf{x}_k$$

into the previous equation. This leads to the following relationship for the next costate at node $k-1$:

$$\boldsymbol{\xi}^{+}_{k-1} = \mathcal{V}_{\mathbf{x}_k} + \mathcal{V}_{\mathbf{x}\mathbf{x}_{k+1}}\delta\mathbf{x}_k$$

with

$$\begin{aligned}\mathcal{V}_{\mathbf{x}_k} &= \mathbf{Q}_{\mathbf{x}_k} + \mathbf{\Pi}_k^\top(\mathbf{Q}_{\mathbf{u}\mathbf{u}_k}\boldsymbol{\pi}_k - \mathbf{Q}_{\mathbf{u}_k}) - \mathbf{Q}_{\mathbf{u}\mathbf{x}_k}^\top\boldsymbol{\pi}_k,\\ \mathcal{V}_{\mathbf{x}\mathbf{x}_k} &= \mathbf{Q}_{\mathbf{x}\mathbf{x}_k} + (\mathbf{\Pi}_k^\top\mathbf{Q}_{\mathbf{u}\mathbf{u}_k} - 2\mathbf{Q}_{\mathbf{u}\mathbf{x}_k}^\top)\mathbf{\Pi}_k.\end{aligned}$$

We then inject this *costate relationship* to solve each subproblem backwards in time.

Author contributions

Carlos Mastalli devised the main ideas behind nullspace parametrization, 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 Corbères 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

[1] P.-B. Wieber, "Holonomy and nonholonomy in the dynamics of articulated motion," in *Fast Motions in Biomechanics and Robotics*, 2005.
[2] R. Featherstone, *Rigid Body Dynamics Algorithms*. Berlin, Heidelberg: Springer-Verlag, 2007.
[3] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2015.
[4] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-Body Nonlinear Model Predictive Control Through Contacts for Quadrupeds," *IEEE Robot. Automat. Lett. (RA-L)*, vol. 3, 2018.
[5] C. Mastalli, W. Merkt, G. Xin, J. Shim, M. Mistry, I. Havoutis, and S. Vijayakumar, "Agile Maneuvers in Legged Robots: a Predictive Control Approach," 2022.
[6] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, 1966.
[7] T. Corbères, C. Mastalli, W. Merkt, I. Havoutis, M. Fallon, N. Mansard, T. Flayols, S. Vijayakumar, and S. Tonneau, "Perceptive Locomotion through Whole-Body MPC and Optimal Region Selection," 2023.
[8] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. USA: Cambridge University Press, 2009.
[9] J. Carpentier and N. Mansard, "Analytical Derivatives of Rigid Body Dynamics Algorithms," in *Robotics: Science and Systems (RSS)*, 2018.
[10] A. Herzog, S. Schaal, and L. Righetti, "Structured contact force optimization for kino-dynamic motion generation," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2016.
[11] S. Fahmi, C. Mastalli, M. Focchi, and C. Semini, "Passive Whole-Body Control for Quadruped Robots: Experimental Validation Over Challenging Terrain," *IEEE Robot. Automat. Lett. (RA-L)*, vol. 4, 2019.
[12] E. Dantec, R. Budhiraja, A. Roig, T. Lembono, G. Saurel, O. Stasse, P. Fernbach, S. Tonneau, S. Vijayakumar, S. Calinon, M. Taix, and N. Mansard, "Whole Body Model Predictive Control with a Memory of Motion: Experiments on a Torque-Controlled Talos," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.
[13] S. Katayama and T. Ohtsuka, "Efficient solution method based on inverse dynamics for optimal control problems of rigid body systems," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.
[14] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Inverse Dynamics vs. Forward Dynamics in Direct Transcription Formulations for Trajectory Optimization," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.
[15] T. Erez and E. Todorov, "Trajectory optimization for domains with contacts using inverse dynamics," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2012.
[16] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics," in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2018.
[17] F. Udwadia and R. Kalaba, "A New Perspective on Constrained Motion," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 1992.
[18] C. Mastalli, J. Marti-Saumell, W. Merkt, J. Sola, N. Mansard, and S. Vijayakumar, "A Feasibility-Driven Approach to Control-Limited DDP," *Autom. Robots.*, 2022.
[19] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010, accessed: 2022-10-02.
[20] J. B. J. G. Frison, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems," in *IEEE Int. Conf. Contr. Apps. (CCA)*, 2013.
[21] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.*, vol. 47, pp. 99–131, 2005.
[22] R. H. Byrd, J. Nocedal, and R. A. Waltz, "KNITRO: An integrated package for nonlinear optimization," in *Large Scale Nonlinear Optimization*, 2006.
[23] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, 2006.
[24] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban, "An Interior Algorithm for Nonlinear Optimization That Combines Line Search and Trust Region Steps," *Math. Program.*, 2006.
[25] "Harwell Subroutine Library, AEA Technology, Harwell, Oxfordshire, England. A catalogue of subroutines," http://www.hsl.rl.ac.uk/, 2011, accessed: 2022-10-02.
[26] P.-B. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2006.
[27] ——, "Viability and predictive control for safe locomotion," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2008.
[28] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2018.
[29] G. Bledt and S. Kim, "Extracting Legged Locomotion Heuristics with Regularized Predictive Control," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2020.
[30] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "MPC-based Controller with Terrain Insight for Dynamic Legged Locomotion," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2020.
[31] N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad, "Model Predictive Control With Environment Adaptation for Legged Locomotion," *IEEE Access*, vol. 9, 2021.
[32] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2017.
[33] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for Torque-Controlled Legged Robots," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2019.
[34] T. Corberes, T. Flayols, P.-A. Leziart, R. Budhiraja, P. Soueres, G. Saurel, and N. Mansard, "Comparison of predictive controllers for locomotion and balance recovery of quadruped robots," *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.
[35] S. Katayama and T. Ohtsuka, "Whole-body model predictive control with rigid contacts via online switching time optimization," 2022.
[36] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," in *Proc. on Fast Mot. in Bio. Rob.* Springer Berlin Heidelberg, 2006.
[37] 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 Int. Conf. Rob. Autom. (ICRA)*, 2020.
[38] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, 1972.
[39] 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 International Symposium on System Integrations (SII)*, 2019.

[40] E. Todorov, “Convex and analytically-invertible dynamics with contacts and constraints: Theory and implementation in MuJoCo,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2014.

[41] D. Gabay, “Minimizing a differentiable function over a differential manifold,” *J. Optim. Theory Appl.*, vol. 37, 1982.

[42] U. Frese, “A Framework for Sparse Non-Linear Least Squares Problems on Manifolds,” Ph.D. dissertation, Universität Bremen, 2008.

[43] R. E. Bellman, “The Theory of Dynamic Programming,” *Bull. Amer. Math. Soc*, 1954.

[44] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, “An efficient optimal planning and control framework for quadrupedal locomotion,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2017.

[45] G. H. Golub and C. F. V. Loan, *Matrix computations*, 4th ed. The Johns Hopkins University Press, 2013.

[46] M. Giftthaler and J. Buchli, “A projection approach to equality constrained iterative linear quadratic optimal control,” in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2017.

[47] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization,” in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2012.

[48] J. Nocedal and S. Wright, *Numerical Optimization*, 2nd ed. New York, USA: Springer, 2006.

[49] R. H. Byrd, M. E. Hribar, and J. Nocedal, “An Interior Point Algorithm for Large-Scale Nonlinear Programming,” *SIAM J. Optim.*, vol. 9, 1999.

[50] R. Fletcher, “A modified Marquardt subroutine for non-linear least squares,” Atomic Energy Research Establishment, Tech. Rep., 1971.

[51] T. A. Howell, B. Jackson, and Z. Manchester, “ALTRO: A Fast Solver for Constrained Trajectory Optimization,” in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2019.

[52] S. Kazdadi, J. Carpentier, and J. Ponce, “Equality Constrained Differential Dynamic Programming,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.

[53] W. Jallet, J. Carpentier, and N. Mansard, “Implicit Differential Dynamic Programming,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.

[54] A. Pavlov, I. Shames, and C. Manzie, “Interior Point Differential Dynamic Programming,” *IEEE Trans. Contr. Sys. Tech. (TCST)*, vol. 29, 2021.

[55] L. zhi Liao and C. A. Shoemaker, “Advantages of Differential Dynamic Programming Over Newton’s Method for Discrete-Time Optimal Control Problems,” Cornell University, Tech. Rep., 1992.

[56] M. Posa, C. Cantu, and R. Tedrake, “A direct method for trajectory optimization of rigid bodies through contact,” *The Int. J. of Rob. Res. (IJRR)*, vol. 33, 2014.

[57] H. Dai, A. Valenzuela, and R. Tedrake, “Whole-body motion planning with centroidal dynamics and full kinematics,” in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2014.

[58] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, “Hierarchical planning of dynamic movements without scheduled contact sequences,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2016.

[59] S. Dafarra, G. Romualdi, and D. Pucci, “Dynamic Complementarity Conditions and Whole-Body Trajectory Optimization for Humanoid Robot Locomotion,” *IEEE Trans. Robot. (T-RO)*, 2022.

[60] R. Deits and R. Tedrake, “Footstep planning on uneven terrain with mixed-integer convex optimization,” in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2014.

[61] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernandez-Lopez, and C. Semini, “Simultaneous Contact, Gait, and Motion Planning for Robust Multilegged Locomotion via Mixed-Integer Convex Optimization,” *IEEE Robot. Automat. Lett. (RA-L)*, vol. 3, 2018.

[62] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, “SL1M: Sparse L1-norm Minimization for contact planning on uneven terrain,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2020.

[63] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, “Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2021.

[64] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, “Learning, planning, and control for quadruped locomotion over challenging terrain,” *The Int. J. of Rob. Res. (IJRR)*, vol. 30, no. 2, 2011.

[65] C. Mastalli, M. Focchi, I. Havoutis, A. Radulescu, S. Calinon, J. Buchli, D. G. Caldwell, and C. Semini, “Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2017.

[66] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, “Robust Rough-Terrain Locomotion with a Quadrupedal Robot,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2018.

[67] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, “Motion Planning for Quadrupedal Locomotion: Coupled Planning, Terrain Mapping and Whole-Body Control,” *IEEE Trans. Robot. (T-RO)*, vol. 36, 2020.

[68] P. Fankhauser, M. Bloesch, C. Gehring, M. Hutter, and R. Siegwart, “Robot-centric elevation mapping with uncertainty estimates,” in *Int. Conf. on Climb. and Walk. Rob. and the Supp. Techn. for Mob. Mach. (CLAWAR)*, 2014.

[69] F. Risbourg, T. Corbères, 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/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2022.

[70] J.-L. Blanco, “A tutorial on SE(3) transformation parameterizations and on-manifold optimization,” University of Malaga, Tech. Rep., 2010.

[71] D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter, “Perception-less terrain adaptation through whole body control and hierarchical optimization,” in *IEEE Int. Conf. Hum. Rob. (ICHR)*, 2016.

[72] R. W. Brockett, “Asymptotic stability and feedback stabilization,” in *Differential Geometric Control Theory*, 1983, pp. 181–191.

[73] T. Boaventura, C. Semini, J. Buchli, M. Frigerio, M. Focchi, and D. G. Caldwell, “Dynamic torque control of a hydraulic quadruped robot,” in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2012.

[74] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, “ANYmal - a highly mobile and dynamic quadrupedal robot,” in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2016.

Carlos Mastalli received an M.Sc. degree in mechatronics engineering from the Simón Bolívar 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 Robotarium 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 Zürich (Switzerland), and the University of Edinburgh (UK). His research focuses on building athletic intelligence for 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 di Ricerca “Enrico Piaggio”, Università di 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.