A Direct-Indirect Hybridization Approach to Control-Limited DDP

Carlos Mastalli, Wolfgang Merkt, Josep Marti-Saumell
Henrique Ferrolho, Joan Solà, Nicolas Mansard, Sethu Vijayakumar

Abstract—**Differential Dynamic Programming (DDP) is an indirect method for trajectory optimization. Its efficiency derives from the exploitation of temporal structure (inherent to optimal control problems) and explicit roll-out/integration of the system dynamics. However, it suffers from numerical instability and, when compared to direct methods, it has limited initialization options (allows initialization of controls, but not of states) and lacks proper handling of control constraints. These limitations are due to the fact that DDP is a single shooting algorithm. In this work, we tackle these issues with a direct-indirect hybridization approach that is primarily driven by the dynamic feasibility of the optimal control problem. Our feasibility search emulates the numerical resolution of a direct transcription problem with only dynamics constraints, namely a multiple shooting formulation. We show that our approach has better numerical convergence than BOX-DDP (a shooting method), and that its convergence rate and runtime performance are competitive with state-of-the-art direct transcription formulations solved using the interior point and active set algorithms available in KNITRO. We further show that our approach decreases the dynamic feasibility error monotonically—as in state-of-the-art nonlinear programming algorithms. We demonstrate the benefits of our hybrid approach by generating complex and athletic motions for quadruped and humanoid robots.**

Index Terms—**optimal control, differential dynamic programming, feasibility, direct and indirect methods**

I. Introduction

OPTIMAL control is a powerful tool to synthesize motions and controls through task goals (cost/optimality) and constraints (e.g., system dynamics, interaction constraints). We can formulate such problems using *direct methods* [1], which first discretize over both state and controls, and then optimize using sparse general-purpose Nonlinear Programming (NLP) libraries such as SNOPT [2], KNITRO [3], and IPOPT [4]. To ensure that state evolves as described by the equations of motion, we define equality constraints in the NLP problem. Algorithms for nonlinear programming can only guarantee the constraint satisfaction at their convergence. When this occurs, we say that the discretized states are *dynamically* feasible. Furthermore, despite the advantage of using advanced libraries for nonlinear programming, these algorithms perform very large matrix factorizations during the computation of the search direction, i.e., the resolution of the Karush-Kuhn-Tucker (KKT) problem. These expensive factorizations limit their application domain to control on reduced models (e.g., [5], [6], [7], [8]) or motion planning (e.g., [9], [10], [11], [12], [13]) in robotics. Furthermore, classical line search methods are less effective than the nonlinear roll-out of the dynamics as it increases the number of iterations (cf. [14]).

Indirect methods, which first optimize over the controls

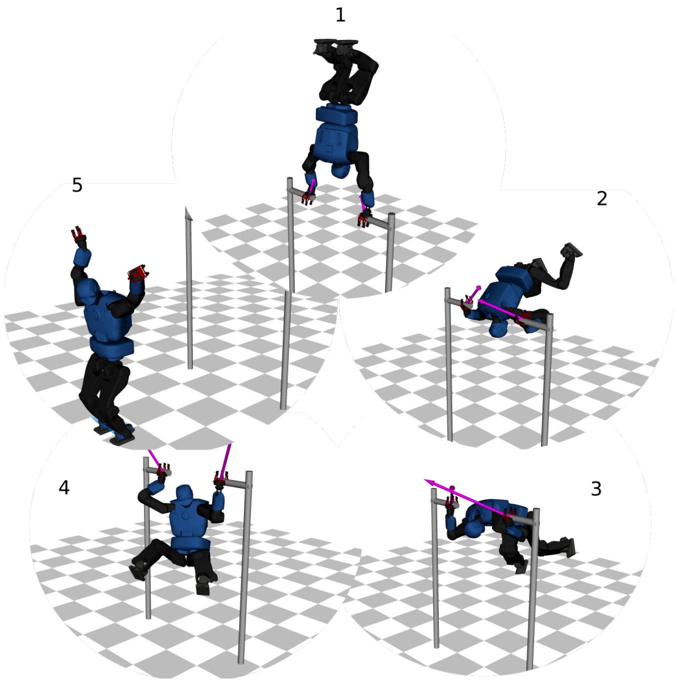

Fig. 1. Snapshots of an athletic behavior computed by our direct-indirect hybridization approach: BOX-FDDP. The optimized motion considers the robot's full-body dynamics, joint limits, and friction cone constraints. We initialize our algorithm using the default posture and quasi-static torques (as described in the results section). We describe three desired actions: grasping the bar at a specific point, raising the feet up-ward a given location (Seq. 1), and landing over a specific placement (Seq. 5). The shown behavior emerges without further specification from our problem formulation. The arrows describe the magnitude and direction of the contact forces. To watch the video, click the figure or see `https://youtu.be/gHcxBjLM9jk`.

This research was supported by (1) the European Commission under the Horizon 2020 project Memory of Motion (MEMMO, project ID: 780684), (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. *(Corresponding author: Carlos Mastalli.)*

Carlos Mastalli and Sethu Vijayakumar are with the School of Informatics, University of Edinburgh, U.K., and the Alan Turing Institute (e-mail: carlos.mastalli@ed.ac.uk; sethu.vijayakumar@ed.ac.uk).

Wolfgang Merkt is with the Oxford Robotics Institute, University of Oxford, U.K. (e-mail: wolfgang@robots.ox.ac.uk).

Josep Marti-Saumell and Joan Solà are with the Institut de Robòtica i Informàtica Industrial, Universitat Politècnica de Catalunya, Spain (e-mail: jmarti@iri.upc.edu; jsola@iri.upc.edu).

Henrique Ferrolho is with the School of Informatics, University of Edinburgh, U.K. (e-mail: henrique.ferrolho@ed.ac.uk.)

Nicolas Mansard is with LAAS-CNRS, France (e-mail: nicolas.mansard@laas.fr).

and then discretize, have once again attracted attention due to recent results on fast nonlinear model predictive control based on DDP (e.g., [15], [16], [17], [18]). In particular, there is a significant interest in the iterative Linear-Quadratic Regulator (iLQR) algorithm [19] as its Gauss-Newton (GN) approximation reduces the computation time while having super-linear convergence. Both iLQR and DDP algorithms solve a Riccati equation in the backward pass, which is obtained by applying the Pontryagin's Maximum Principle (PMP). The computation of the derivatives of the Value function in the backward pass represents the *costate integration*; instead, the roll-out of the system dynamics describes the *state integration* step. In contrast to direct methods, these indirect approaches exploit the temporal/Markovian structure of the optimal control problem by solving a sequence of smaller sub-problems that are obtained through Bellman's principle of optimality [20]. This leads to fast and cheap computations due to very small matrix factorizations and effective data cache accesses. Despite these advantages, both algorithms are unable to handle equality and inequality constraints efficiently. Furthermore, they have poor globalization capability (i.e., require a good initialization in order to converge) and are prone to numerical instability—commonly recognized challenges for single shooting approaches [1]. These undesirable properties are mainly due to the fact that the iLQR/DDP algorithms implicitly enforce the dynamic feasibility through the system roll-out.

A. Related work

Trade-offs between feasibility and optimality appear in most of the state-of-the-art nonlinear programming libraries. For instance, IPOPT includes a feasibility restoration phase which aims at reducing the constraint violation [4]. In KNITRO, the progress on both feasibility and optimality is achieved by adding an ℓ^1-norm penalty term for the constraints in the *merit* function [3]. In fact, by changing the merit function or the line search procedure, we can put emphasis on obtaining feasible solutions *before* trying to optimize them. Instead, the iLQR/DDP algorithms do not make this trade-off as the backward and forward passes, in their classical form, do not accept infeasible iterations. However, recent work on *direct-indirect* hybridization [21], [22] has provided ways of handling dynamically infeasible iterations, which we elaborate below.

The direct-indirect hybridization approaches in [21], [22] are rooted in dynamic programming. For instance, Giftthaler et al. [21] introduced a *lifted*[1] version of the Riccati equation that allows initialization of both state and control trajectories; it further accounts for the relinearization required by the direct-indirect hybridization in the backward pass. In turn, in our previous work [22], we proposed a modification of the forward pass that numerically matches the gap contraction expected by a *direct transcription* method subject to equality constraints only. It constructs the adjoint and control equation (Riccati sweep) using the PMP and defines the behavior of the defect constraints based on the First-order Necessary Condition (FONC) of optimality.[2] These hybrid approaches improve numerical robustness against poor initialization, as they are able to use an initial guess for the state trajectory. Unfortunately, none of these methods handle inequality constraints such as control limits, with the exception of a recent work that computes squashed control sequences [25].

There are two main strategies for incorporating arbitrary constraints: active set and penalization methods (as extensively described in [24]). In the robotics community, one of the first successful attempts to incorporate inequality constraints in DDP used an active set approach [26], which is based on pioneering work in the control community [27]. Concretely, this approach focused on handling control limits during the computation of the backward pass, i.e., in the minimization of the *control Hamiltonian*.[3] The method is popularly named BOX-DDP, and the authors also showed a better convergence rate when compared with a squashing function approach. Later, Xie et al. [28] included general inequality constraints into the control Hamiltonian and the forward pass. The method sacrifices the computational performance by including a second quadratic program, which is solved in the forward pass. However, it still remains faster than solving the same problem using direct collocation with SNOPT as reported in [29].

Generally speaking, active set methods are suitable for small-size problems (such as minimizing control Hamiltonian problem described above) as their accuracy and speed often outperform other methods. However, they are not safe for control due to their lack of tight bounds. This motivates the development of penalty-based methods, despite their numerical difficulties: ill-conditioning and slow convergence. To overcome these difficulties, Lantoine and Russell [30] proposed a method that incorporates an augmented Lagrangian term. This method was studied in the context of robust thrust optimization, in which the dynamical system has fewer degrees of freedom compared to complex legged robots. Later, Howell et al. [31] extended the augmented Lagrangian approach to handle arbitrary inequality constraints for aerial navigation and manipulation problems. Additionally, the algorithm incorporates an active set projection for solution polishing and is often faster than direct collocation solved with IPOPT or SNOPT.

Our work bridges the gap between direct and indirect methods for nonlinear optimal control problems with control limits. Our main motivation for proposing this hybridization is to increase the algorithm's basins of attraction, by focusing on feasibility instead of focusing solely on efficiency and optimality. Apart from the control limits and dynamics, we handle all remaining constraints (e.g., state and friction cone) through quadratic penalization, as described in the results section.

B. Contribution

The main contribution of this work is the first complete study of the numerical properties, behaviors, and guarantees of direct-indirect hybridization in differential dynamic programming. It relies on four technical contributions:

[1]This name is coined by [23], and we refer to *gaps* or *defects* produced between multiple shooting nodes.

[2]For more details about the FONC of optimality see [24].

[3]In the following section we formally describe the control Hamiltonian.

(i) an original and efficient optimal control algorithm that directly handles control limits (BOX-FDDP),
(ii) a mathematical proof for the existence of feasible descent directions,
(iii) extensive comparisons against direct transcription and BOX-DDP, and
(iv) an experimental validation of the dynamic feasibility evolution against interior point and active set algorithms for nonlinear programming.

Our hybridization approach builds on top of our previous results on feasibility-driven search [22], for which we hereby propose to define two modes in our algorithm: feasibility-driven and control-bounded. It also goes beyond the iterative Gauss-Newton shooting methods proposed by [21], as it considers the dynamic feasibility in the forward pass and explicitly incorporates control limits. Additionally, our approach has outstanding numerical capabilities, which allow us to generate motions that go beyond state-of-the-art methods on optimal control or trajectory optimization in robotics, e.g., the athletic maneuver of a humanoid robot shown in Fig. 1.

II. DIRECT METHODS AND DIFFERENTIAL DYNAMIC PROGRAMMING

Before describing our hybridization approach, we introduce direct transcription, and explain its numerical advantages when compared to indirect methods such as DDP (Section II-A). Then, in Section II-B, we describe the salient aspects of BOX-DDP with an original connection to indirect methods. This section contains known material, although we believe it contributes (i) to unveil the underlying problems of differential dynamic programming, and (ii) to understand the theoretical foundations of our direct-indirect hybridization.

A. Direct transcription for optimal control

We consider a direct transcription approach for the nonlinear optimal control problem with control bounds:

$$\begin{aligned}
\min_{\mathbf{x}_s,\mathbf{u}_s} \quad & \ell_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} \ell_k(\mathbf{x}_k,\mathbf{u}_k) \\
\text{s.t.} \quad & \bar{\mathbf{f}}_0 \quad := \mathbf{x}_0 \ominus \tilde{\mathbf{x}}_0 = \mathbf{0}, \\
& \bar{\mathbf{f}}_{k+1} := \mathbf{f}(\mathbf{x}_k,\mathbf{u}_k) \ominus \mathbf{x}_{k+1} = \mathbf{0}, \qquad \forall k=\{0,1,\cdots,N-1\} \\
& \underline{\mathbf{u}} \leq \mathbf{u}_k \leq \bar{\mathbf{u}}, \qquad \forall k=\{0,1,\cdots,N-1\}
\end{aligned} \tag{1}$$

where the state $\mathbf{x} \in X$ lies in a differential manifold (with dimension n_x); the control $\mathbf{u} \in \mathbb{R}^{n_u}$ defines the input commands; $\underline{\mathbf{u}}$, $\bar{\mathbf{u}}$ are the lower and upper control bounds; $\tilde{\mathbf{x}}_0$ is the initial state of the system; $\ominus$ describes the *difference* operator of the state manifold; $N \in \mathbb{N}$ describes the number of nodes (horizon); $\ell_N(\cdot)$, $\ell_k(\cdot)$ are the terminal and running cost functions; and $\bar{\mathbf{f}}_0$, $\bar{\mathbf{f}}_{k+1} \in T_{\mathbf{x}}X$ are the gap residual functions that impose the dynamic feasibility; and $T_{\mathbf{x}}X$ describes the tangent space of the state manifold at the state $\mathbf{x}$.

Eq. (1) describes a nonlinear program as the system dynamics are transcribed into a set of algebraic equations. As depicted in Fig. 2, the transcription process incorporates state and control trajectories $(\mathbf{x}_s,\mathbf{u}_s)$ as decision variables. This is in contrast to many indirect methods, such as differential dynamic programming, which only transcribe the control sequence $\mathbf{u}_s$ and obtain $\mathbf{x}_s$ by integrating the system dynamics.

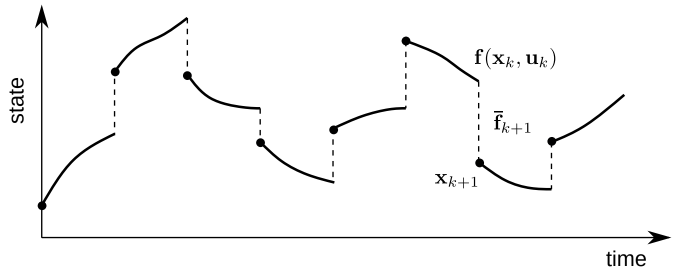

Fig. 2. A schematic of a direct transcription formulation. Different intermediate states are introduced as decision variables. A set of equality constraints enforce the dynamics feasibility. The black dots represent the initial states used for the system roll-out. The dashed lines represent the gaps between the roll-outs. Dynamic feasibility is achieved once these gaps are closed.

1) Numerical behavior of direct transcription: Algorithms for nonlinear programming aim at finding the Karush-Kuhn-Tucker (KKT) conditions defined by the FONC of optimality [24]. This process involves iteratively solving a KKT problem (i.e., linear system of equations) until satisfaction of a stopping criterion. In the *line search* strategy, the solution of this KKT problem provides a search direction $\delta\mathbf{w}_k$, and the selected step length α defines how much the current guess $\mathbf{w}_k$ moves along that direction, i.e., $\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha\delta\mathbf{w}_k$. If we apply this approach to direct transcription problems without control bounds, then the KKT problem for a single shooting interval k is

$$\overbrace{\begin{bmatrix} \boldsymbol{\ell}_{\mathbf{xx}_k} & \boldsymbol{\ell}_{\mathbf{xu}_k} \\ \boldsymbol{\ell}_{\mathbf{xu}_k}^T & \boldsymbol{\ell}_{\mathbf{uu}_k} \end{bmatrix}}^{\nabla^2\boldsymbol{\Phi}_k} \overbrace{\begin{bmatrix} \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}}^{\delta\mathbf{w}_k} + \overbrace{\begin{bmatrix} \mathbf{I} & -\mathbf{f}_{\mathbf{x}_k}^T \\ & -\mathbf{f}_{\mathbf{u}_k}^T \end{bmatrix}}^{\nabla\mathbf{g}_k} \overbrace{\begin{bmatrix} \boldsymbol{\lambda}_k^+ \\ \boldsymbol{\lambda}_{k+1}^+ \end{bmatrix}}^{\boldsymbol{\Lambda}_k^+} = -\overbrace{\begin{bmatrix} \boldsymbol{\ell}_{\mathbf{x}_k} \\ \boldsymbol{\ell}_{\mathbf{u}_k} \end{bmatrix}}^{\nabla\boldsymbol{\Phi}_k}, \tag{2}$$

$$\begin{bmatrix} \mathbf{I} & \\ -\mathbf{f}_{\mathbf{x}_k} & -\mathbf{f}_{\mathbf{u}_k} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix} = \overbrace{\begin{bmatrix} \bar{\mathbf{f}}_k \\ \bar{\mathbf{f}}_{k+1} \end{bmatrix}}^{\mathbf{g}_k}, \tag{3}$$

where Eq. (2), (3) are the dual and primal feasibility of the FONC of optimality, respectively; $\delta\mathbf{x}_k$, $\delta\mathbf{u}_k$ define the search direction; $\boldsymbol{\lambda}_k^+$, $\boldsymbol{\lambda}_{k+1}^+$ are the updated Lagrangian multipliers; $\boldsymbol{\ell}_{\mathbf{x}_k}$, $\boldsymbol{\ell}_{\mathbf{u}_k}$, and $\boldsymbol{\ell}_{\mathbf{xx}_k}$, $\boldsymbol{\ell}_{\mathbf{xu}_k}$, $\boldsymbol{\ell}_{\mathbf{uu}_k}$ are the Jacobians and Hessians of the cost function; and $\mathbf{f}_{\mathbf{x}_k}$, $\mathbf{f}_{\mathbf{u}_k}$ are the Jacobians of the system dynamics. Note that we apply the Gauss-Newton (GN) approximation as we ignore the Hessian of the system dynamics to avoid expensive tensor-vector multiplications.

When we factorize this system of equations, the resulting search direction always satisfies the dynamics constraints if the Jacobians and Hessians are constant (i.e., a LQR problem). However, if we apply an α-step, the gap of the dynamics closes by a factor of $(1-\alpha)$. We observe this by inspecting the primal feasibility at the next iteration:

$$\begin{aligned}
\bar{\mathbf{f}}_{k+1}^{i+1} &= \bar{\mathbf{f}}_{k+1}^{i} - (\delta\mathbf{x}_{k+1} - \mathbf{f}_{\mathbf{x}k}\delta\mathbf{x}_k - \mathbf{f}_{\mathbf{u}k}\delta\mathbf{u}_k) \\
&= (1-\alpha)(\mathbf{f}(\mathbf{x}_k^i,\mathbf{u}_k^i) - \mathbf{x}_{k+1}^i),
\end{aligned} \tag{4}$$

where, by definition in Eq. (3), we have that

$$\alpha\bar{\mathbf{f}}_{k+1}^{i} = \delta\mathbf{x}_{k+1} - \mathbf{f}_{\mathbf{x}k}\delta\mathbf{x}_k - \mathbf{f}_{\mathbf{u}k}\delta\mathbf{u}_k,$$

with the gap defined as $\bar{\mathbf{f}}^i_{k+1} = \mathbf{f}(\mathbf{x}^i_k, \mathbf{u}^i_k) - \mathbf{x}^i_{k+1}$, and i describes the iteration number.

2) Advantages of direct transcription: The rationale for a direct transcription approach (namely, adding $\mathbf{x}_s$ as decision variables) is to distribute the nonlinearities of the dynamics over the entire horizon [32]. To illustrate this statement, we recognize that integrating over a horizon implies recursively calling *integrator* functions, i.e.,

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \\ &= \mathbf{f}(\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}), \mathbf{u}_k) \\ &= \vdots \\ &= \mathbf{f}(\mathbf{f}(\cdots \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0), \mathbf{u}_{k-1}), \mathbf{u}_k), \end{aligned} \tag{5}$$

in which the nonlinearity increases along the horizon. This means that the local prediction, constructed by the derivatives of the nonlinear system in the KKT problem, becomes more inaccurate as the horizon increases. This is a well recognized drawback of single shooting formulations, and certainly a numerical limitation of the BOX-DDP algorithm described below.

B. Differential dynamic programming with control limits

As proposed by [26], BOX-DDP locally approximates the optimal flow (i.e., the Value function V) as

$$\begin{aligned} V_k(\delta\mathbf{x}) = \min_{\delta\mathbf{u}_k} \ell_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) + V_{k+1}(\mathbf{f}_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k)), \\ \text{s.t.} \qquad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}} \ , \end{aligned} \tag{6}$$

which breaks the optimal control problem into a sequence of simpler sub-problems with control bounds. Then, a local search direction is computed through a Linear Quadratic (LQ) approximation of the Value function:

$$\begin{aligned} &\delta\mathbf{u}^*_k(\delta\mathbf{x}_k) = \\ &\arg\min_{\delta\mathbf{u}_k} \overbrace{\frac{1}{2}\begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}^T \begin{bmatrix} 0 & \mathbf{Q}^T_{\mathbf{x}_k} & \mathbf{Q}^T_{\mathbf{u}_k} \\ \mathbf{Q}_{\mathbf{x}_k} & \mathbf{Q}_{\mathbf{xx}_k} & \mathbf{Q}_{\mathbf{xu}_k} \\ \mathbf{Q}_{\mathbf{u}_k} & \mathbf{Q}^T_{\mathbf{xu}_k} & \mathbf{Q}_{\mathbf{uu}_k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k \end{bmatrix}}^{\mathbf{H}(\delta\mathbf{x}_k, \delta\mathbf{u}_k, \bar{V}_k, k)}, \\ &\text{s.t.} \qquad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}}, \end{aligned} \tag{7}$$

where the $\mathbf{Q}_k$ terms describe the LQ approximation of the Hamiltonian function $\mathbf{H}(\cdot)$, and the derivatives of the Value function $\bar{V}_k = (V_{\mathbf{x}_k}, V_{\mathbf{xx}_k})$ take the role of the costate variables.[4] Solving Eq. (7) results in a local feedback control law $\delta\mathbf{u}_k = \mathbf{k}_k + \mathbf{K}_k\delta\mathbf{x}_k$ consisting of a feed-forward term $\mathbf{k}_k$ and a state feedback gain $\mathbf{K}_k$ for each discretization point k. Below, we describe how to compute the $\mathbf{Q}_k$ terms in the so-called Riccati sweep step.

1) Riccati sweep: The LQ approximation of the Hamiltonian is computed recursively, backwards in time, as follows

$$\begin{aligned} \mathbf{Q}_{\mathbf{x}_k} &= \boldsymbol{\ell}_{\mathbf{x}_k} + \mathbf{f}^T_{\mathbf{x}_k} V_{\mathbf{x}_{k+1}}, \\ \mathbf{Q}_{\mathbf{u}_k} &= \boldsymbol{\ell}_{\mathbf{u}_k} + \mathbf{f}^T_{\mathbf{u}_k} V_{\mathbf{x}_{k+1}}, \\ \mathbf{Q}_{\mathbf{xx}_k} &= \boldsymbol{\ell}_{\mathbf{xx}_k} + \mathbf{f}^T_{\mathbf{x}_k} V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{x}_k}, \\ \mathbf{Q}_{\mathbf{xu}_k} &= \boldsymbol{\ell}_{\mathbf{xu}_k} + \mathbf{f}^T_{\mathbf{x}_k} V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{u}_k}, \\ \mathbf{Q}_{\mathbf{uu}_k} &= \boldsymbol{\ell}_{\mathbf{uu}_k} + \mathbf{f}^T_{\mathbf{u}_k} V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{u}_k}, \end{aligned} \tag{8}$$

[4]For more details about the costate variables see [33].

where $V_{\mathbf{x}_{k+1}}$, $V_{\mathbf{xx}_{k+1}}$ are obtained by solving the following Riccati equations at $k+1$:

$$\begin{aligned} V_{\mathbf{x}_{k+1}} &= \mathbf{Q}_{\mathbf{x}_{k+1}} - \mathbf{Q}_{\mathbf{xu}_{k+1}} \mathbf{Q}^{-1}_{\mathbf{uu}_{k+1}} \mathbf{Q}_{\mathbf{u}_{k+1}}, \\ V_{\mathbf{xx}_{k+1}} &= \mathbf{Q}_{\mathbf{xx}_{k+1}} - \mathbf{Q}_{\mathbf{xu}_{k+1}} \mathbf{Q}^{-1}_{\mathbf{uu}_{k+1}} \mathbf{Q}^T_{\mathbf{xu}_{k+1}}. \end{aligned} \tag{9}$$

Additionally, we use the gradient and Hessian of the Value function to find a local search direction as described below.

2) Control-bounded direction: We compute the control-bounded direction, defined in Eq. (7), by breaking it down into the so-called *feed-forward* and *feedback* sub-problems [26]. We first compute the feed-forward term by solving the following Quadratic Programming (QP) program with box constraints:

$$\begin{aligned} &\mathbf{k}_k = \arg\min_{\delta\mathbf{u}_k} \frac{1}{2}\delta\mathbf{u}^T_k \mathbf{Q}_{\mathbf{uu}_k} \delta\mathbf{u}_k + \mathbf{Q}^T_{\mathbf{u}_k}\delta\mathbf{u}_k, \\ &\text{s.t.} \qquad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}}, \end{aligned} \tag{10}$$

and then the feedback gain as

$$\mathbf{K}_k = -\hat{\mathbf{Q}}^{-1}_{\mathbf{uu},f_k} \mathbf{Q}_{\mathbf{ux}_k}, \tag{11}$$

where $\hat{\mathbf{Q}}^{-1}_{\mathbf{uu},f_k}$ is the control Hessian of the free subspace obtained in Eq. (10) in which f_k describes the index of the free subspace at node k.

We efficiently solve the BOX-QP program (feed-forward sub-problem) by using the Projected-Newton QP algorithm [34] as it quickly identifies the active set and moves along the free subspace of the Newton step. With this algorithm, we also get a similar computational cost to the unconstrained QP if the active set remains unchanged. Thus, the runtime performance is similar to the DDP algorithm. However, it requires a feasible warm-start $\delta\mathbf{u}^0_k$.

By using a Projected-Newton QP algorithm, we further efficiently obtain the control Hessian of the free subspace $\hat{\mathbf{Q}}^{-1}_{\mathbf{uu},f_k}$ as the algorithm computes it internally when it moves along the free subspace of the Newton step. With it, we compute a state feedback gain that generates corrections within the control limits. This is an important feature for controlling the robot as well as for rolling-out the nonlinear dynamics in the forward pass.

3) State integration: In any DDP algorithm such as BOX-DDP, we perform a state integration using the locally-linear policy as

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbf{x}_0, \\ \hat{\mathbf{u}}_k &= \mathbf{u}_k + \alpha\mathbf{k}_k + \mathbf{K}_k(\hat{\mathbf{x}}_k \ominus \mathbf{x}_k), \quad {\scriptstyle \forall k=\{0,1,\cdots,N-1\}} \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{f}(\hat{\mathbf{x}}_k, \hat{\mathbf{u}}_k), \quad {\scriptstyle \forall k=\{0,1,\cdots,N-1\}} \end{aligned} \tag{12}$$

where $\hat{\mathbf{x}}_k$, $\hat{\mathbf{u}}_k$ are the new state and control at node k generated using a step length α. The feedback gain helps to distribute the nonlinearities; however, as seen in the previous section, it does not resemble the numerical behavior described by the FONC of optimality in direct transcription. This different numerical behavior stems from the state integration procedure closing the gaps, i.e., $\bar{\mathbf{f}}_k = \mathbf{0}$, $\forall k = \{0, 1, \cdots, N\}$.

4) Costate integration: Solving the Riccati equations, described in Eq. (9), is equivalent to performing the costate integration procedure as $V_{\mathbf{x}_{k+1}}$ and $V_{\mathbf{xx}_{k+1}}$ can be seem as a LQ approximation of the costate at the node k. Furthermore, the expected improvement

$$\Delta V_k = -\frac{1}{2}\mathbf{Q}^T_{\mathbf{u}_k}\mathbf{Q}^{-1}_{\mathbf{uu}_k}\mathbf{Q}_{\mathbf{u}_k} \tag{13}$$

is equivalent to the rate of change of the costate. Below, we elaborate the proposed algorithm based on the aforementioned description.

III. BOX-FDDP: A DIRECT-INDIRECT HYBRIDIZATION APPROACH

We now introduce a novel algorithm that combines a direct-indirect hybridization (Section II-A) with an active set treatment of the control limits (Section II-B) named BOX-FDDP. The BOX-FDDP algorithm comprises two modes: *feasibility-driven* and *control-bounded* modes, that one of which is chosen for a given iteration (Algorithm 1). The feasibility-driven[5] mode uses a direct-indirect hybridization to compute the search direction and step length (lines 4-10 and 11-21, respectively). This mode has a similar numerical behavior to direct transcription, and it neglects the control limits of the system as its focuses on dynamic feasibility only. In contrast, the control-bounded mode projects the search direction onto the feasible control region whenever the dynamics constraint is feasible (line 10). In both modes, the applied controls in the forward pass are projected onto their feasible box (line 13), causing dynamically-infeasible iterations to reach the control box. With this strategy, our solver focuses on feasibility early on, which increases its basins of attraction, and later on optimality. Technical descriptions of both modes are elaborated in Sections III-A and III-B. Note that the formal guarantees for the existence of feasible descent directions are introduced later in Section III-C.

A. Search direction

In the standard BOX-DDP algorithm, an initial forward pass is performed to obtain the initial state trajectory $\mathbf{x}_s$. This trajectory enforces the dynamics implicitly; thus, the gaps are zero, i.e., $\bar{\mathbf{f}}_k = \mathbf{0}$ for all $k = \{0, 1, \cdots, N-1\}$. Instead, our hybridized variant, BOX-FDDP, computes the gaps once at each iteration (line 3), which are used to find the search direction and to compute the expected improvement. However, if the iteration is dynamically feasible, then the search direction procedure is the same as in the standard BOX-DDP. Below, we describe the steps performed to compute the search direction.

1) Computing the gaps: Given a current iterate $(\mathbf{x}_s, \mathbf{u}_s)$, we perform a nonlinear roll-out to compute the gaps as

$$\bar{\mathbf{f}}_{k+1} := \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \ominus \mathbf{x}_{k+1}, \quad \forall k=\{0,1,\cdots,N-1\} \tag{14}$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is the roll-out state at interval $k+1$, $\mathbf{x}_{k+1}$ is the next shooting state, and $\ominus$ is the difference operator.

[5]Here, *feasibility* concerns the dynamics of the system, not the feasibility of other problem constraints.

Algorithm 1: Control-limited FDDP (BOX-FDDP)

```
1  compute LQ approximation of the cost and dynamics
2  if infeasible iterate then
3      compute the gaps, Eq. (14)
4  for $k \leftarrow N-1$ to $0$ do
5      update the feasibility-driven Hamiltonian, Eq. (17)
6      if infeasible iterate then
7          compute feasibility-driven direction, Eq. (19)
8      else
9          project BOX-QP warm start, Eq. (20)
10         compute control-bounded
             direction, Eq. (10)-(11)
11 for $\alpha \in \{1, \frac{1}{2}, \cdots, \frac{1}{2^n}\}$ do
12     for $k \leftarrow 0$ to $N$ do
13         project control onto the feasible box, Eq. (21)
14         if infeasible iterate or $\alpha \neq 1$ then
15             update the gaps, Eq. (22)
16         else
17             close the gaps,
                 $\mathbf{f}_k = \mathbf{0}\ \forall k \in \{0, \cdots, N-1\}$
18         perform step, Eq. (23)
19     compute the expected improvement, Eq. (24)
20     if success step then
21         break
```

2) Hamiltonian of direct transcription formulation: Without loss of generality, we use the Gauss-Newton (GN) approximation [19] to write the Hamiltonian function of the standard DDP algorithm as

$$\begin{aligned}\mathbf{H}(\cdot) = &\frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_{k+1}\end{bmatrix}^T \begin{bmatrix}0 & V^T_{\mathbf{x}_{k+1}}\\ V_{\mathbf{x}_{k+1}} & V_{\mathbf{xx}_{k+1}}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_{k+1}\end{bmatrix}\\ &+\frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix}^T\begin{bmatrix}0 & \boldsymbol{\ell}^T_{\mathbf{x}_k} & \boldsymbol{\ell}^T_{\mathbf{u}_k}\\ \boldsymbol{\ell}_{\mathbf{x}_k} & \boldsymbol{\ell}_{\mathbf{xx}_k} & \boldsymbol{\ell}_{\mathbf{xu}_k}\\ \boldsymbol{\ell}_{\mathbf{u}_k} & \boldsymbol{\ell}^T_{\mathbf{xu}_k} & \boldsymbol{\ell}_{\mathbf{uu}_k}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix}\end{aligned}, \tag{15}$$

where again $\boldsymbol{\ell}_{\mathbf{x}_k}, \boldsymbol{\ell}_{\mathbf{u}_k}$ and $\boldsymbol{\ell}_{\mathbf{xx}_k}, \boldsymbol{\ell}_{\mathbf{xu}_k}, \boldsymbol{\ell}_{\mathbf{uu}_k}$ describe the gradient and Hessian of the cost function, respectively; $\delta\mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x}_k}\delta\mathbf{x}_k + \mathbf{f}_{\mathbf{u}_k}\delta\mathbf{u}_k$ is the linearized dynamics; and $\mathbf{f}_{\mathbf{x}_k}, \mathbf{f}_{\mathbf{u}_k}$ are its Jacobians.

In direct transcription settings, linearization of the system dynamics includes a drift term

$$\delta\mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x}k}\delta\mathbf{x}_k + \mathbf{f}_{\mathbf{u}k}\delta\mathbf{u}_k + \bar{\mathbf{f}}_{k+1}, \tag{16}$$

as there are gaps in the dynamics term $\bar{\mathbf{f}}_{k+1}$ produced between subsequent shooting segments (q.v. Fig. 2). Then, according to the Pontryagin's Maximum Principle (PMP), the Riccati sweep needs to be adapted as follows:

$$\begin{aligned}\mathbf{Q}_{\mathbf{x}_k} &= \boldsymbol{\ell}_{\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{x}_{k+1}}^+,\\ \mathbf{Q}_{\mathbf{u}_k} &= \boldsymbol{\ell}_{\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{x}_{k+1}}^+,\\ \mathbf{Q}_{\mathbf{xx}_k} &= \boldsymbol{\ell}_{\mathbf{xx}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{x}_k},\\ \mathbf{Q}_{\mathbf{xu}_k} &= \boldsymbol{\ell}_{\mathbf{xu}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{u}_k},\\ \mathbf{Q}_{\mathbf{uu}_k} &= \boldsymbol{\ell}_{\mathbf{uu}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{xx}_{k+1}} \mathbf{f}_{\mathbf{u}_k}\end{aligned} \tag{17}$$

in which

$$V_{\mathbf{x}_{k+1}}^+ = V_{\mathbf{x}_{k+1}} + V_{\mathbf{xx}_{k+1}} \bar{\mathbf{f}}_{k+1} \tag{18}$$

is the gradient of the Value function after the deflection produced by $\bar{\mathbf{f}}_{k+1}$. Note that the Hessian of the Value function remains unchanged as DDP approximates the Value function through a LQ model. Additionally, this modification affects the values of the Riccati equations, Eq. (9), and costate rate, Eq. (13), as they depend on the gradient of the Value function.

3) Feasibility-driven direction: For *dynamically-infeasible* iterates (line 7), we ignore the control constraints and compute a *control-unbounded* direction:

$$\begin{aligned}\mathbf{k}_k &= -\mathbf{Q}_{\mathbf{uu}_k}^{-1} \mathbf{Q}_{\mathbf{u}_k},\\ \mathbf{K}_k &= -\mathbf{Q}_{\mathbf{uu}_k}^{-1} \mathbf{Q}_{\mathbf{ux}_k}.\end{aligned} \tag{19}$$

We do this because we cannot quantify the effect of the gaps on the control bounds, which are needed to solve the feed-forward sub-problem Eq. (10). Our approach is equivalent to opening the control bounds during dynamically-infeasible iterates.

4) Control-bounded direction: We warm-start the BOX-QP using the feed-forward term $\mathbf{k}_k$ computed in the previous iteration. However, if the algorithm is switching from feasibility to control-bounded mode (i.e., the previous iteration is infeasible), then $\mathbf{k}_k$ might fall outside the feasible box and $\underline{\mathbf{u}} - \mathbf{u}_k \leq \mathbf{k}_k \leq \bar{\mathbf{u}} - \mathbf{u}_k$ do not hold. This violates the assumption of the previously-described BOX-QP, for which a feasible initial point needs to be provided.

To handle infeasible iterates, we propose to project the warm-start of the BOX-QP (line 9) as

$$[\![\mathbf{k}_k]\!]_{\underline{\mathbf{u}},\bar{\mathbf{u}}} = \min\left(\max\left(\mathbf{k}_k, \underline{\mathbf{u}} - \mathbf{u}_k\right), \bar{\mathbf{u}} - \mathbf{u}_k\right), \tag{20}$$

where $\underline{\mathbf{u}}$, $\bar{\mathbf{u}}$ are the lower and upper bounds of the feed-forward sub-problem, Eq. (10), respectively.

Once we project the warm-start $\mathbf{k}_k$, we solve the feed-forward and feedback sub-problems as explained in Section II-B2. Furthermore, as described above, we solve the BOX-QP using a Projected-Newton method [34], which handles box constraints efficiently.

B. Step length

As far as we know, the standard BOX-DDP [26] modifies only the search direction (i.e., backward pass) to handle the control limits. However, it is also important to project the controls onto the feasible box during the forward pass. We do this by finding a step length that minimizes the cost [24].

1) Projecting the roll-out towards the feasible box: We propose to project the controls onto the feasible box in the nonlinear roll-out (line 13), i.e.,

$$\hat{\mathbf{u}}_k \leftarrow \min\left(\max\left(\hat{\mathbf{u}}_k, \underline{\mathbf{u}}\right), \bar{\mathbf{u}}\right), \tag{21}$$

where $\hat{\mathbf{u}}_k$ is the updated control from the control policy. Our method does not require to solve another QP problem [28] or to project the linear search direction given the gaps on the dynamics [31]. Furthermore, the control policy considers a gap prediction that guarantees a feasible descent direction. We formally describe the technical details of this procedure in Section III-B3.

2) Updating the gaps: As analyzed earlier, the evolution of the gaps in direct transcription is affected by the selected step length. For an optimal control problem without control limits, this evolution is defined as

$$\bar{\mathbf{f}}_k \leftarrow (1-\alpha)\bar{\mathbf{f}}_k, \tag{22}$$

where α is the step-length found by the line-search procedure (line 11-21). Note that a full step ($\alpha = 1$) closes the gaps completely. We described this gap contraction rate in Section II-A1.

3) Nonlinear step: With a nonlinear roll-out[6] (line 18), we avoid the linear prediction error of the dynamics that is typically handled by a *merit* function in off-the-shelf NLP algorithms, as explained in [22]. If we keep the gap-contraction rate of Eq. (22), then we obtain

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

where $\hat{\mathbf{x}}_k$, $\hat{\mathbf{u}}_k$ are the tried state and control along an α-step; $\mathbf{k}_k$ and $\mathbf{K}_k$ are the feed-forward term and feedback gains computed by Eq. (19) or Eq. (10)-(11). Furthermore, the initial condition of the roll-out is defined as $\hat{\mathbf{x}}_0 = \tilde{\mathbf{x}}_0 - (1-\alpha)\bar{\mathbf{f}}_0$. Note that this is in contrast to the standard BOX-DDP, in which the gaps are always closed, even for $\alpha < 1$.

Avoiding the use of a merit function helps the algorithm to check the search direction more accurately. Indeed, it has been shown that the nonlinear roll-out is more effective than a standard line search procedure as it reduces the number of iterations [14].

4) Expected improvement: It is critical to properly evaluate the success of a trial step. Given the current dynamics gaps $\bar{\mathbf{f}}_k$, BOX-FDDP computes the expected improvement of a computed search direction as

$$\Delta J(\alpha) = \Delta_1\alpha + \frac{1}{2}\Delta_2\alpha^2, \tag{24}$$

with

$$\begin{aligned}\Delta_1 &= \sum_{k=0}^{N} \mathbf{k}_k^\top \mathbf{Q}_{\mathbf{u}_k} + \bar{\mathbf{f}}_k^\top (V_{\mathbf{x}_k} - V_{\mathbf{xx}_k}\hat{\mathbf{x}}_k),\\ \Delta_2 &= \sum_{k=0}^{N} \mathbf{k}_k^\top \hat{\mathbf{Q}}_{\mathbf{uu},f_k} \mathbf{k}_k + \bar{\mathbf{f}}_k^\top (2V_{\mathbf{xx}_k}\mathbf{x}_k - V_{\mathbf{xx}_k}\bar{\mathbf{f}}_k),\end{aligned} \tag{25}$$

[6]In this work, a nonlinear *roll-out* is also referred to as a nonlinear step.

where $\hat{\mathbf{Q}}_{\mathbf{uu},f_k}$ is the control Hessian of the free space, and J is the total cost of a given state-control trajectory $(\mathbf{x}_s, \mathbf{u}_s)$. We use this expected improvement model for both modes. Note that, in the feasibility-driven mode, the free space spans the entire control space; instead, in the control-bounded mode, the gaps are zero.

We obtain this expression by computing the cost from a linear roll-out of the current control policy as described in Eq. (23). We also accept ascent directions since we use the Goldstein condition to check for the trial step. Ascent directions improve the algorithm convergence as it helps to reduce the feasibility error through a moderate increment in the cost. For more details about the Goldstein condition see [24].

5) Regularization: We regularize the $\mathbf{Q_{uu}}$ and $\mathbf{V_{xx}}$ terms through a Levenberg-Marquardt scheme [35]. Concretely, we increase the damping value μ when the computation of the feed-forward sub-problem—formulated in equation (10)—fails, or when the forward pass accepts a step length smaller than $\alpha_0 = 0.01$. Moreover, we decrease the damping value if the iteration accepts a step larger than $\alpha_1 = 0.5$. Both regularization procedures modify the values of the $\mathbf{Q_{uu}}$ and $\mathbf{V_{xx}}$ terms during the Riccati sweep computation as

$$\begin{aligned} \mu' &\leftarrow \beta_{i,d}\mu, \\ \mathbf{Q_{uu}} &\leftarrow \mathbf{Q_{uu}} + \mu'\mathbf{I}, \\ \mathbf{V_{xx}} &\leftarrow \mathbf{V_{xx}} + \mu'\mathbf{I}, \end{aligned}$$

where β_i and β_d are the factors[7] used to increase or decrease the current damping value μ, respectively; μ' is the newly-computed damping value; and $\mathbf{I}$ is the identity matrix. Additionally, we start the regularization procedure with an initial, and user-defined, damping value.[8] We also define minimum and maximum damping values to avoid increasing or decreasing unnecessary the damping value.[9]

The $\mathbf{Q_{uu}}$ regularization significantly increases the robustness of the algorithm and ensures convergence, as it moves from Newton direction to steepest-descent, and vice versa. The Newton direction, which occurs with $\mu = 0$, provides fast convergence and is robust against scaling because it exploits the Hessian of the problem. However, it does not always produce valid descent directions as $\mathbf{Q_{uu}}$ might be negative-define and the problem nonconvex. In such cases, increasing the damping value ensures that $\mathbf{Q_{uu}}$ is positive-define which, in turn, computes a search direction closer to the steepest-descent one. Adding regularization on $\mathbf{V_{xx}}$ enforces the state trajectory to be closer to the one previously computed [36]. This regularization on $\mathbf{V_{xx}}$ does not change the search direction as in the $\mathbf{Q_{uu}}$ regularization and, crucially, will not result in vanishing feedback gains even for large damping values.

C. Existence of feasible descent directions

As described above, our approach has two main modes: feasibility-driven and control-bounded. During the feasibility-driven phase, we compute a search direction to drive the next guess towards dynamic feasibility and try a step while keeping the control within the box constraints. This projection procedure can be seen as a nonlinear term in our dynamics, but we assume its effect is negligible for finding a feasible direction. On the other hand, our algorithm computes a search direction that considers the box constraints after the dynamic feasibility has been achieved. This is needed to improve the next current guess by taking control constraints into account when computing the feedback gains along the free subspace.

The feasibility-driven direction is computed by mimicking the numerical behavior of a nonlinear program during the resolution of a direct transcription problem with only dynamics constraints as analyzed in Section II-A1. It implies that our feasibility-driven search produces a descent direction, and eventually the algorithm converges, if the cost Hessian $\nabla^2\mathbf{\Phi}_s$ is a positive definite matrix. Indeed, the positiveness is always guaranteed by our regularization procedure as described before. Furthermore, the feasibility-driven step aims at reducing the nonlinearities produced by a single shooting formulation. When the dynamics are feasible, we apply a control-bounded search which also produces a descent direction as it boils down to a QP program.

In the next section, we present a series of results that demonstrate the benefits of our hybridization approach.

IV. RESULTS

We analyze BOX-FDDP across a wide range of optimal control problems, which are briefly introduced in Section IV-A, as follows. First, in Section IV-B, we show the benefits of the feasibility mode by analyzing the gap contraction and its connection with the nonlinearities in the dynamics. Second, we compare our algorithm against a direct transcription formulation in Section IV-C. Concretely, we compare the dynamic feasibility and optimality evolutions, runtime performance, and robustness to different initial guesses against the interior point and active set algorithms available in KNITRO. Finally, in Section IV-D, we report the results of a squashing approach for solving the control bounds as it demonstrates the numerical performance of having two modes.

We benchmark the algorithms using an Intel Core i9-9900KF CPU with eight cores @ 3.60GHz and 16 MB cache. Our implementation of BOX-FDDP supports code generation, but all runtime performances reported henceforth do not employ it for fair comparison with other approaches. We use 8 threads to compute the cost and dynamics derivatives for the experiments with the BOX-FDDP algorithm and the legged robots only. We use the same initial regularization and stopping criteria values for each problem, and the values are 10^{-9} and 5×10^{-5}, respectively.

A. Optimal control problems

To provide substantial evidence on the benefits of the direct-indirect hybridization approach, we developed a range of different optimal control problems: 1) an under-actuated double pendulum (PEND); 2) a quadcopter navigating towards a goal (GOAL) or through a narrow passage (NARROW) and

[7] $\beta_{i,d}$ commonly range between 2–10. In this work, we set $\beta_{i,d} = 10$.

[8] In this work, we use 10^{-9} as initial regularization value.

[9] We use 10^{-16} and 10^{12} as the minimum and maximum damping values, respectively. Note that 10^{-16} is approximately the resolution of a `double` number.

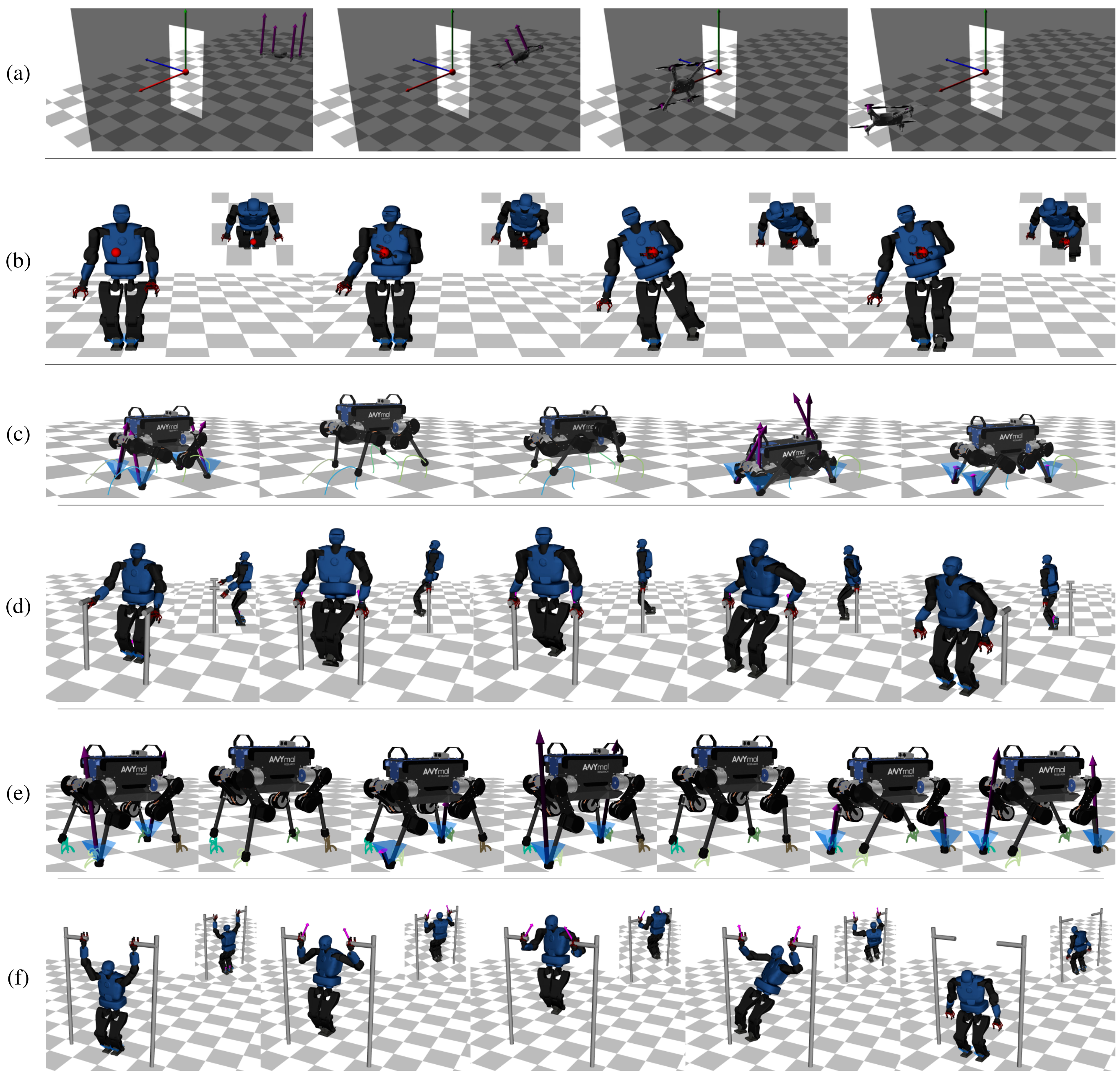

Fig. 3. Snapshots of different robot maneuvers computed by BOX-FDDP. (a) traversing a narrow passage with a quadcopter (GOAL); (b) Talos balancing on a single leg (TAICHI); (c) aggressive jumping of 30 cm that reaches ANYmal limits (JUMP); (d) Talos dipping on a parallel bars (DIP); (e) ANYmal hopping with two legs (HOP); (f) Talos performing a pull-up bar task (PULLUP). To watch the video, click its respective figure or see `https://youtu.be/gHcxBjLM9jk`.

looping (LOOP); 3) various gaits, aggressive jumps (JUMP) and unstable hopping (HOP) in a quadruped robot; 4) whole-body manipulation (MAN), hand control while balancing in single leg (TAICHI), dip on parallel bars (DIP) and a pull-up bar task (PULLUP) in a humanoid robot. Fig. 3 shows snapshots of motions computed by BOX-FDDP for some of these problems, and the accompanying video shows the entire motion sequences.[10]

We describe the cost functions, dynamics, control limits, penalization terms, and initialization of each optimal control problem in Appendix A. For a given experiment, any modification of those specifications will be clearly stated in its respective section. Finally, some of these problems, as well as our implementation of the BOX-FDDP algorithm, are publicly available in the CROCODDYL repository [37].

[10]Supplementary video: `https://youtu.be/gHcxBjLM9jk`

B. *Advantages of the feasibility-driven mode*

To understand the benefits of the feasibility-driven mode, we analyze the resulting total cost, number of iterations and total computation time obtained in both algorithms: BOX-FDDP and BOX-DDP^{+}. BOX-DDP^{+} is an improved version of the standard BOX-DDP described in [26]. This version accepts infeasible warm-starts as in Eq. (17), and it is available in the CROCODDYL repository. Without this modification, the standard BOX-DDP could easily diverge (and not converge at

TABLE I
NUMBER OF ITERATIONS, TOTAL COST, AND AVERAGE TOTAL COMPUTATION TIME OVER 100 TRIALS.

	Box-DDP$^+$			Box-FDDP (feas)		
Problems	Iter.	Cost	Time (s.)	Iter.	Cost	Time (s.)
PEND	105	4.4292	0.2473	34	0.4997	0.0721
GOAL	23	0.0764	0.0913	18	0.0072	0.0775
LOOP	133	6.7211	0.9144	56	0.6444	0.3982
NARROW	70	1.9492	0.6781	35	0.4577	0.3136
MAN	71	4.6193	11.428	66	4.6193	7.7417
TAICHI	120	6.8184	38.482	101	6.8184	21.909
JUMP	108	1.34e5	✗	53	6.67e4	0.8226
HOP	17	1.3e12	✗	205	1.91e4	19.844
DIP	121	34.2	106.9	97	34.2	107.8
PULLUP	176	276.87	✗	426	146.29	422.5

✗ algorithm does not find a solution.

all) when we initialize it using quasi-static torques in problems with medium to longer horizons.

1) Greater globalization strategy and convergence: In our experiments, Box-DDP$^+$ was unable to generate jumping and hopping motions for quadrupeds, as well as pull-ups for humanoids, i.e., it failed to solve our JUMP, HOP, and PULLUP task specifications (marked by the ✗ in Table I). Box-DDP$^+$ failed to generate such aggressive motions because trajectories satisfying all the specifications of those tasks are significantly distant from the initial guess provided to the solver and, as previously mentioned, Box-DDP$^+$ behaves poorly when initialized with a guess sufficiently far away from a locally-optimal solution.

In contrast, our approach (Box-FDDP) was able to solve all of the tasks, and it did so with fewer iterations and lower total cost (see Table I and Fig. 4). Furthermore, Box-FDDP and Box-DDP$^+$ have the same algorithmic complexity, but since our approach requires fewer iterations than Box-DDP$^+$, the total computation time of our approach is lower. These results are a direct consequence of the feasibility-driven mode in our approach, which is able to find control sequences even when faced with poor initial guesses. The infeasible iterations are part of a globalization strategy, which ensures convergence from remote initial guesses through a balance between optimality and feasibility.

Finally, we would like to emphasize that the feasibility-driven mode of Box-FDDP can not only solve tasks that Box-DDP$^+$ is unable to solve, but also *improve* the solutions of tasks that Box-DDP$^+$ is able to solve. For instance, consider the quadcopter tasks GOAL, NARROW, and LOOP: In the accompanying video, we show that our approach generates concise and smooth quadcopter trajectories, whereas Box-DDP$^+$ generates jerky motions and with unnecessary loops, due to early projection of the control commands.

2) Gap contraction and nonlinearities: We observed that the gap contraction rate is highly influenced by the nonlinearities of the system dynamics (see Fig. 5). When compared to the dynamics, the nonlinearities of the task often have a smaller effect (e.g., DIP vs PULLUP). Indeed, the gap contraction speed followed the order: humanoid, quadruped, double pendulum, and quadcopter.

Propagation errors due to the dynamics linearization have an important effect on the algorithm progress as DDP-based methods maintain a local quadratic approximation of the Value function. The prediction of the expected improvement is indeed more accurate for systems with less nonlinearities, which is the reason why the algorithm tends to accept larger steps that result in higher gap contractions. Indeed, the effect of having a feasibility search is more significant in problems with very nonlinear dynamics as it reduces the total cost faster due to the nonlinearity distribution (see Fig. 4 and 5). It also produces a low cost reduction rate during the gap contraction

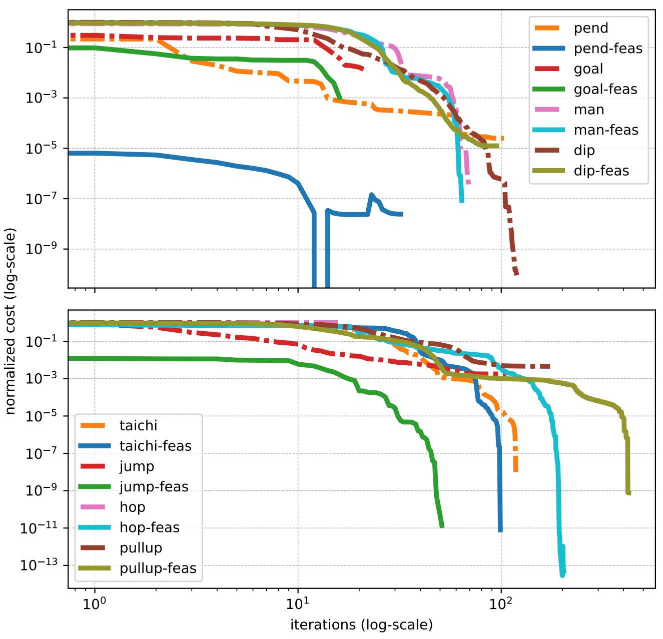

Fig. 4. Cost and convergence comparison for different optimal control problems. Box-FDDP outperformed Box-DDP$^+$ in all the cases: (top) double pendulum (PEND), quadcopter navigation (QUAD), and whole-body manipulation (MAN), Talos dipping on parallel bars (DIP); and (bottom) whole-body balance (TAICHI), quadrupedal jumping (JUMP), quadrupedal hopping (HOP), Talos' pull-up workout (PULLUP). Box-FDDP (*-feas) solved the problem with fewer iterations and often with lower cost than Box-DDP$^+$. Furthermore, Box-DDP$^+$ failed to solve some of the hardest problems: i.e., quadrupedal jumping and hopping, Talos' pull-up task. Our algorithm showed greater globalization, i.e., it is less sensitive to poor initialization compared with Box-DDP$^+$.

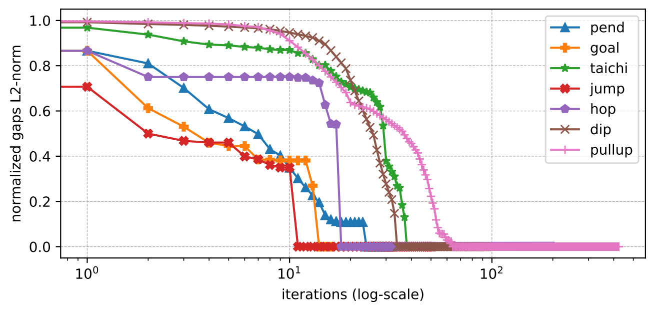

Fig. 5. Gap contraction of Box-FDDP for different optimal control problems. For all the cases, the gaps were open for the first several iterations. The gap contraction rate varies according to the accepted step-length. Smaller contraction rates, during the first iterations, appeared in very nonlinear problems (TAICHI, MAN, HOP, and JUMP), because of the larger error of the search direction.

phase as the algorithm is first focusing on achieving dynamic feasibility.

3) Highly-dynamic and complex maneuvers: The Box-FDDP algorithm can solve a wide range of motions: from unstable and consecutive hops to aggressive and complex motor behaviors. In Fig. 6, we show the joint torques and velocities of a single leg for the ANYmal's jumping problem (depicted in Fig. 3-c). The motion consisted of three phases: jumping (0–300 ms), flying (300–700 ms), and landing (700–1000 ms). We used 0.7 as a friction coefficient and reduced the real joint limits of the ANYmal robot: from 40 to 32 $\mathrm{N\,m}$ (torque limits) and from 15 to 7.5 rad/s (velocity limits). Thus, generating a 30 cm jump becomes a very challenging task. Furthermore, in this experiment, the velocity limits violations appeared due to we used quadratic penalization (with constant weights) to enforce them and the swing phases are probably short for this large jump. Note that if we use constant weights, it might turn out that, for some cases, these weights are not big enough. Nonetheless, we only encountered these violations in very constrained problems. For instance, we did not find velocity violations for the walking, trotting, pacing, and bounding gaits (reported in the accompanying video). For these cases, Box-FDDP converged approximately with the same number of iterations achieved by the FDDP solver (i.e., a fully unconstrained case). This is due to the fact that the robot could generate those gaits without reaching its torque limits.

Surprisingly, naturally looking behaviors emerged during the computation of the DIP and PULLUP problems on the Talos humanoid robot. We did not include any heuristic that could have helped the algorithm to generate these undefined behaviors. For instance, the balancing and leg-crossing on the bars emerge if we allocate a significant amount of time in that motion phase. Similarly, the pull-up motion emerges if we significantly increase the maximum torque limits on the arms.

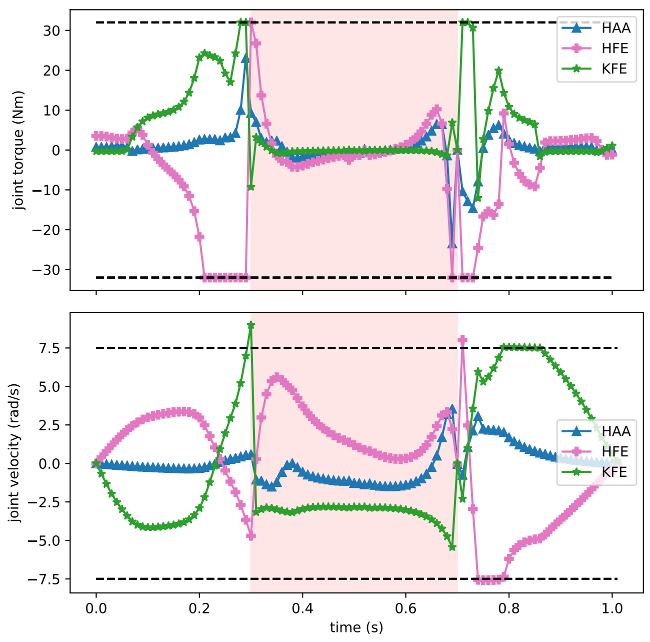

Fig. 6. Joint torques and velocities for the ANYmal jumping maneuver. (top) Generated torques of the LF joints and its limits (32 $\mathrm{N\,m}$); (bottom) Generated velocities of the LF joint and its limits (7.5 rad/s). The red region describes the flight phase. Note that HAA, HFE, and KFE are the abduction/adduction, hip flexion/extension and knee flexion/extension joints, respectively.

C. *Box-FDDP vs a direct transcription formulation*

We formulated an efficient direct transcription problem with dynamics defects constraints in each node. The formulation is conceptually similar to our feasible descent direction introduced in Section III-C. We transcribed the dynamics using a symplectic Euler scheme, the same integration scheme also used in Box-FDDP. We solved the direct transcription problem using different optimization algorithms provided by KNITRO [3]. These available algorithms are: Interior/Direct [38], Interior/CG [39], Active Set [40] and Sequential Quadratic Programming (SQP) algorithms. Below we briefly describe each of KNITRO's algorithms, and then report the comparison results with Box-FDDP.

The Interior/Direct (KNITRO-IDIR) algorithm replaces the NLP problem with a series of barrier sub-problems. In each iteration, it solves the primal-dual KKT problem using a line search procedure.[11] Instead, Interior/CG (KNITRO-ICG) solves the primal-dual KKT problem using a projected conjugate gradient method. This method uses exact second derivatives, without explicitly storing the Hessian matrix, through a tailored trust region procedure. Interior/Direct also invokes this trust region procedure if the line search iteration converges to a non-stationary point [38]. In contrast to the interior point methods, the Active Set algorithm (KNITRO-SLQP) replaces the NLP problem with a sequence of quadratic programs to form a sequential linear-quadratic programming algorithm. This algorithm selects a set of active constraints in each iteration, and produces a more exterior path (i.e., along the constraints) to the solution. Finally, KNITRO's SQP algorithm (KNITRO-SQP) is also an active set method designed for small to medium scale problems with expensive function evaluations. Both active set approaches are often preferable to interior point methods on small- to medium-sized problems when we can provide a good initial guess [24]. However, the problems in robotics are often large with many inequality constraints. Indeed, the benefits of interior point methods have been pointed out in the context of direct methods [6].

1) Optimality vs feasibility: We compared the total cost, number of iterations, and total computation time against the different algorithms implemented in KNITRO over 100 trials. For the comparison, we solved the double pendulum problem (PEND), as it requires discovery of a swing-up maneuver. With this, we can clearly compare the trade-off between optimality and feasibility across the different algorithms. Note that, as described earlier, we used a single-thread for both KNITRO and Box-FDDP despite that our algorithm supports multithreading.

Table II reports three different formulations used in the

[11]For more details about interior point methods, the authors suggest the reader to see [24].

TABLE II
BOX-FDDP VS KNITRO IN DOUBLE PENDULUM PROBLEM.

Case	*Algorithms*	Cost	Iteration	Total Time (sec.)
	BOXFDDP	0.4997	34	**0.0721**
PEN	KNITRO-IDIR	2.1094	1064	8.6704
REGCONST	KNITRO-IDIR	**0.4693**	47	0.2766
CONST	KNITRO-IDIR	–	20	0.1387
PEN	KNITRO-ICG	0.5441	59	0.2390
REGCONST	KNITRO-ICG	0.4787	50	0.1875
CONST	KNITRO-ICG	–	**16**	0.0733
PEN	KNITRO-SLQP	0.5978	165	1.2595
REGCONST	KNITRO-SLQP	0.4773	287	1.6704
CONST	KNITRO-SLQP	–	27	0.2359
PEN	KNITRO-SQP	0.5889	114	46.88
REGCONST	KNITRO-SQP	0.4767	182	12.233
CONST	KNITRO-SQP	–	22	8.7736

KNITRO algorithms. The first one (PEN) emulates exactly the optimal control formulation used in BOX-FDDP, i.e., control constraints, regularization terms, and a terminal quadratic cost. The second case (REGCONST) uses a terminal constraint to impose the desired up-ward position together with the regularization terms. The third case (CONST) uses only the terminal constraints. Below we summarize the obtained results for each formulation.

BOX-FDDP converges faster (w.r.t. time) than KNITRO algorithms in all of the above formulations. However, KNITRO-ICG is as fast as our approach with the CONST formulation. On the other hand, when it comes to optimality, KNITRO produces more optimal solutions if we use the REGCONST formulation. Indeed, in our experience, KNITRO generally has a better behavior when the formulation is dominated by constraint functions. Note that we do not report the cost values for the CONST formulation as this boils down to a feasibility problem, i.e., a problem with only constraints.

We used the REGCONST formulation to be able to compare both: cost and feasibility evolution. The dynamic infeasibility decreased monotonically for all the algorithms as plotted in Fig. 7 (top). BOX-FDDP shows a fast resolution of the dynamics feasibility as interior point algorithms that, generally speaking, require less iterations than active set approaches [41]. The cost evolution is also similar to the interior point algorithms, where the total costs are reported as REGCONST cases in Table II.

2) Computation time: BOX-FDDP had a better runtime performance than the KNITRO algorithms for the double pendulum problem (cf. Table II). However, to answer the runtime performance scalability to higher-dimensional optimal control problems, we analyzed the problem of generating a forward jumping maneuver with the ANYmal robot (i.e., JUMP).

We used the same phase timings, joint limits and friction coefficient reported in Section IV-B3. The results reported with KNITRO and BOX-FDDP cases are based on slightly different optimal control formulations. The idea is to define the most suitable formulation for each algorithm. For instance, we use quadratic penalization terms to impose the desired foothold placement, joint velocity limits and friction cone

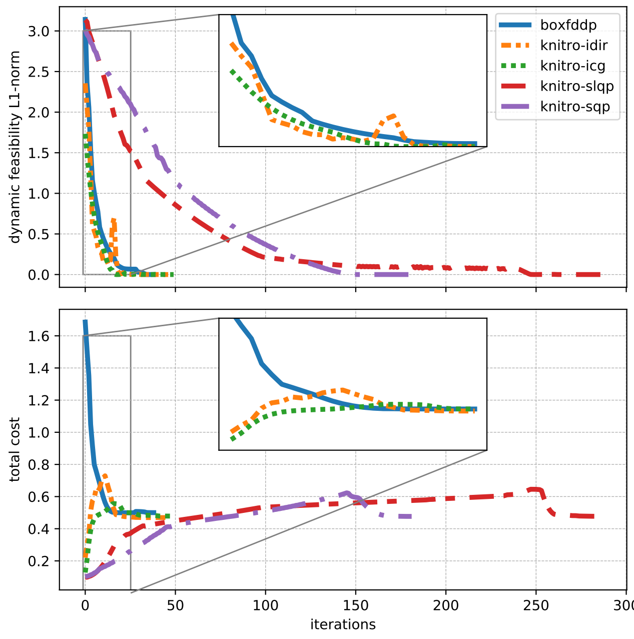

Fig. 7. Dynamic feasibility and cost evolution for the double pendulum problem. BOX-FDDP had a similar evolution (dynamic feasibility and cost) to the interior point algorithms implemented in KNITRO. KNITRO produced lower cost solutions, but the computational burden is often much higher.

TABLE III
RUNTIME PERFORMANCE FOR A FORWARD JUMPING MANEUVER OVER 100 TRIALS.

	KNITRO-IDIR		BOX-FDDP	
Trunk height (m.)	Iter.	Time (sec.)	Iter.	Time (sec.)
0.25	**11**	1.5039	59	**0.9061**
0.30	**10**	1.3900	72	**1.1409**
0.35	**15**	1.9565	59	**0.9030**
0.40	**18**	2.3864	49	**0.7849**
0.45	**16**	2.1320	51	**0.8574**

constraints for the BOX-FDDP algorithm. Instead, for the KNITRO algorithms, we substitute these penalization terms by general equality and inequality constraints. To further reduce the computation time of KNITRO cases, we also impose a constraint for the terminal position of the trunk. Note that we did not include any cost term since it negatively affects the convergence rate of KNITRO, i.e., we treated it as a feasibility problem.

Table III reports the runtime performance over 100 trials for the KNITRO-IDIR algorithm only. The other methods (i.e., KNITRO-ICG, KNITRO-SLQP and KNITRO-SQP) were unable to solve this problem. We used 5 different initial trunk heights, and we initialized the algorithms using their corresponding joint posture (as described in Appendix A-C) and no controls (i.e., $\mathbf{u}_s^0 = \{\mathbf{0}, \cdots, \mathbf{0}\}$). As in the double pendulum case, BOX-FDDP also solved this problem faster than KNITRO algorithms, even though it required a significant number of extra—computationally inexpensive—iterations. We suspect that this increment in the number of iterations is due to the use of penalization terms in the contact placement, friction cone and state limits constraints.

3) Robustness against different initial guesses: We compared the robustness against different initial guesses for the double pendulum (PEND) and quadrupedal jump (JUMP) problems. In both problems, we generated random joint postures—around the nominal state—and used them to define an initial guess for the state trajectory $\mathbf{x_s}^0$. In addition to the robot's joint postures, we also generated random joint velocities around the *zero-velocity* condition for the double pendulum case only. We used this single random posture and velocity for each node in $\mathbf{x_s}^0$, and initialized the control sequence with zeros. We used the most suitable formulations for BOX-FDDP and KNITRO algorithms as justified above.

Table IV reports the number of successful resolutions over 100 trials. We considered a problem to be successfully solved if the gradient of the BOX-FDDP or the feasibility of the KNITRO algorithms are lower than 5×10^{-5} (absolute feasibility tolerance). Note that this includes the cases where KNITRO found a feasible approximate solution.[12] Furthermore, we considered a problem resolution unsuccessful if the problem does not converge within 70 s, which is enough time as we can see above. For each problem, we used two different maximum values of the random initialization, which their maximum magnitude are described using the ℓ^∞ norm (i.e., $\|\cdot\|_\infty$).

As expected, the KNITRO interior point methods performed better than the active set ones. For the double pendulum problem, the interior point algorithms (i.e., KNITRO-IDIR and KNITRO-ICG) perform better than BOX-FDDP if the warm-starting point is close to the initial condition. Despite that, BOX-FDDP shows more robustness to initial guesses as its percentage of successful resolutions is consistent. Furthermore, we observed a significant increment in the number of successful resolutions for the JUMP problem. Indeed, KNITRO was not able to solve this problem at all for random magnitudes bigger than $\|0.01\|_\infty$.

D. Box-FDDP, Box-DDP, and squashing approach in nonlinear problems

To evaluate the numerical performance of having two modes, we compared the BOX-FDDP (with two modes depending on the dynamics feasibility), BOX-DDP$^+$ (using a single mode) and DDP$^+$ with a squashing function (using a single mode) for three scenarios with the IRIS quadcopter: reaching goal (GOAL), looping maneuver (LOOP), and traversing a narrow passage (NARROW). We used a sigmoidal element-wise squashing function of the form:

$$\mathbf{s}^i(\mathbf{u}^i) = \frac{1}{2}\left(\underline{\mathbf{u}}^i + \sqrt{\gamma^2 + (\mathbf{u}^i - \underline{\mathbf{u}}^i)^2}\right) + \frac{1}{2}\left(\overline{\mathbf{u}}^i - \sqrt{\gamma^2 + (\mathbf{u}^i - \overline{\mathbf{u}}^i)^2}\right)$$

in which the sigmoid is approximated through two smooth-abs functions, γ defines its smoothness, and $\underline{\mathbf{u}}^i$, $\overline{\mathbf{u}}^i$ are the element-wise lower and upper control bounds, respectively. We introduced this squashing function on the system controls as: $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{s}(\mathbf{u}_k))$. We used $\gamma = 2$ for all the experiments presented in this section.

Fig. 8 shows that BOX-FDDP converged faster than the other approaches. As reported in the accompanying video, BOX-FDDP did not generate undesired loops and jerky motions as in the other cases. Indeed, the solutions with BOX-FDDP have the lowest cost values (cf. Table I). We also observed that the squashing approach often converges sooner compared to BOX-DDP$^+$. The main reason is due to the early saturation of the controls performed by BOX-DDP$^+$.

In Fig. 9, we show the cost evolution for 10 different initial configurations of the reaching goal task. The target and initial configurations are $(3, 0, 1)$ and $(-0.3 \pm 0.6, 0, 0)$ m, respectively. Infeasible iterations, in BOX-FDDP, produce a very low cost in the first iterations. The squashing approach is the most sensitive to initial conditions. However, on average, it

[12]For further detail, we suggest the reader to consult the KNITRO manual: https://www.artelys.com/docs/Knitro/3_referenceManual.html.

TABLE IV
PERCENTAGE OF SUCCESSFUL RESOLUTIONS FROM RANDOM INITIAL GUESSES.

	PEND		JUMP	
Algorithms	$\|1\|_\infty$	$\|100\|_\infty$	$\|0.0005\|_\infty$	$\|0.005\|_\infty$
BOX-FDDP	74 %	**44** %	**99** %	**99** %
KNITRO-IDIR	**100** %	1 %	51 %	13 %
KNITRO-ICG	**100** %	0 %	✗	✗
KNITRO-SLQP	95 %	0 %	✗	✗
KNITRO-SQP	92 %	0 %	✗	✗

✗ algorithm does not find a solution.

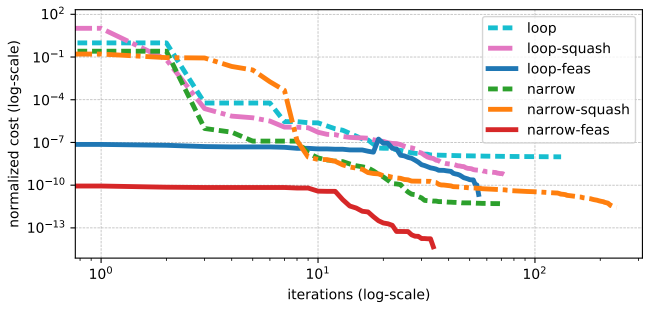

Fig. 8. Cost and convergence comparison for different quadcopter maneuvers: looping (LOOP) and narrow passage traversing (NARROW). BOX-FDDP (*-feas) outperformed both BOX-DDP$^+$ and DDP with squashing function (*-squash).

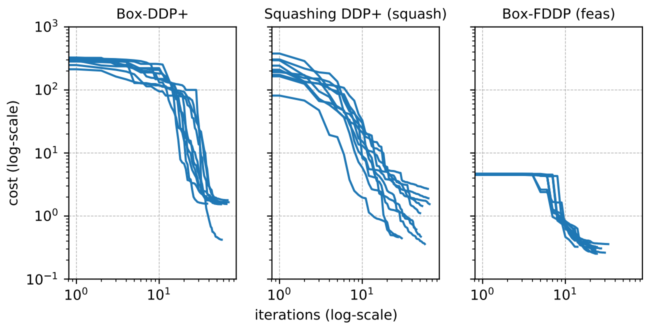

Fig. 9. Costs associated for 10 different initial configurations of reaching goal tasks. BOX-FDDP converges earlier and with lower total cost than BOX-DDP$^+$ and DDP$^+$ with squashing function. The performance of the squashing function approach exhibits a high dependency on the initial condition.

produces slightly better solutions than BOX-DDP$^+$. This is in contrast to the reported results in [26], where the performance was analyzed only for the linear-quadratic regulator problem and without any hybridization.

V. CONCLUSION

We proposed a direct-indirect hybridization approach whose search is primarily driven by the dynamics feasibility of the optimal control problem. The dynamically-infeasible iterations allowed our algorithm to solve a wide range of optimal control problems despite it being provided with a poor initialization. The benefits of the direct-indirect hybridization approach are crystallized over a set of athletic and highly-dynamic maneuvers computed for the Talos humanoid and the ANYmal quadruped robots, respectively. The improvement on the globalization capabilities of our approach has been a key factor to optimize such kind of complex maneuvers while considering the robot's full rigid body dynamics, joint limits and friction cone constraints. Indeed, our method has shown an increment in the robustness against different initial guesses compared with advanced KNITRO algorithms.

We have provided mathematical and experimental evidence that our approach produces descent search directions. For instance, we have observed that the feasibility contraction decreases monotonically as often happens in the advanced nonlinear programming algorithms available in KNITRO. Our approach has also shown to quickly reduce the dynamic infeasibility as observed in the most competitive KNITRO algorithms. A similar effect is observed in the cost evolution as well. Our results suggest that the gap contraction rate is influenced by the nonlinearities of the system dynamics.

The runtime performance of our approach is often superior to direct transcription solved using state-of-the-art KNITRO algorithms despite the increase in number of iterations due to the use of quadratic penalization terms. Indeed, we expect a significant amount of runtime reduction by designing an algorithm that handles general equality and inequality constraints directly. However, when comparing the computation time per iteration, our approach is between 2 to 10 times faster, which makes it suitable for model predictive control applications. One additional remark is that we have not considered the runtime reduction due to code generation support in CROCODDYL via CPPADCODEGEN [42] and CPPAD [43]. According to our experience, code generation can lead to a computation time reduction between $30\,\%$ to $60\,\%$ as can be seen in our public benchmarks [37].

We have developed and reported the results for a wide range of optimal control problems in robotics. These problems cover an important part of the spectrum of robotics applications. The comparison across all these problems is unusual, and it represents an important contribution to the research community since we have open sourced many of these examples, as well as the BOX-FDDP algorithm, in the CROCODDYL repository [37].

ACKNOWLEDGMENTS

The authors are grateful to Matt Timmons-Brown and Vladimir Ivan, from the University of Edinburgh, for the production of the audio material of our video and the fruitful discussions on the study of the effects of our feasibility-driven approach, respectively.

APPENDIX A
OPTIMAL CONTROL PROBLEMS

We divide the optimal control problems into four subsections: double pendulum, quadcopter, quadruped and humanoid robots.

A. Double pendulum (pend)

The goal is to swing from the stable to the unstable equilibrium points, i.e. from down-ward to up-ward positions, respectively. To increase the problem complexity, the double pendulum (with weight of ≈ 4.5 N) has a single actuated joint with small range of control (from -5 to $5\,\mathrm{N\,m}$, largely insufficient for a static execution of the trajectory). The time horizon is $1\,\mathrm{s}$ with 100 nodes. We define a quadratic cost function, for each node, that aims to reach the up-ward position. For the running and terminal nodes, we use the weight values of 2×10^{-4} and 2×10^{4}, respectively. Additionally, we provide state and control regularization terms. To inspect the algorithm capabilities, we do not provide an initial guess, thus the swing-up strategy is discovered by the solver itself. Finally, we implement the cost, dynamics, and their analytical derivatives.

B. Quadcopter

We consider three tasks for the IRIS quadcopter: reaching goal (GOAL), looping maneuver (LOOP), and traversing a narrow passage (NARROW). We define different way-points to describe the tasks, where each way-point specifies the desired pose and velocity. The way-points are described through cost functions in the robot placement and velocity at specific instants of the motion. These cost functions are quadratic term with 10^2 as the weight value for both: pose and velocity. We also include quadratic regularization terms for the state and control, their weight values are 10^{-5} and 10^{-1}, respectively. The vehicle pose is described as a $\mathbb{SE}(3)$ element, which allows us to consider any kind of motion such as *looping* maneuvers. Control inputs are considered to be the thrust produced by the propellers, which can vary within a range from 0.1 to $10.3\,\mathrm{N}$ each. We compute the dynamics using the Articulated Body Algorithm (ABA) algorithm[13], and the analytical derivatives are calculated as described in [45]. We integrate the dynamics with a fixed time step of 10 ms. The solution is computed from a cold-start of the algorithm.

C. Aggressive jump, unstable hopping, and various gaits

We use the ANYmal quadruped robot to generate a wide range of motions—jumping, hopping, walking, trotting, pacing, and bounding. We deliberately reduce the torque and velocity limits to $32\,\mathrm{N\,m}$ and $7.5\,\mathrm{rad/s}$, respectively, which intentionally increases the complexity of the jumping task

[13]For more details about the ABA algorithm see [44].

(JUMP). We use a quadratic barrier to penalize the joint velocities and the contact forces that are outside the limits. The unstable hopping problem (HOP) has a long horizon: $7.14\,\mathrm{s}$ with 714 nodes. It includes 10 hops in total with a phase that switches the feet in contact. Instead, the other problems have a horizon between $0.7{-}1\,\mathrm{s}$ with $70-100$ nodes. We define quadratic terms, with a weight value of 10^6, to track the desired swing-foot placements for each case. We regularize the state trajectory around the robot's default configuration. We use a friction coefficient of 0.7. Furthermore, we formulate a multi-phase optimal control problem using rigid contact dynamics, and their analytical derivatives, as described in [46], [22], respectively. During a contact transition, we employ the impulse dynamics with analytical derivatives as also described in [22]. Our contact dynamics model also includes the Baumgarte gains, which are needed to numerically stabilize differential algebraic equations [47]; we use the values: $(0, 50)$. We initialize the solver using the default posture and the quasi-static torques for each node of the initial guess trajectories. The default posture defines the standing position of the robot; it does not provide any relevant information for a specific maneuver (e.g., JUMP). The quasi-static torques describe the forces required to cancel the effects of gravity subject to the robot's default posture.

D. Whole-body manipulation and balance

We consider four problems for the Talos humanoid robot: whole-body manipulation (MAN), hand control while balancing in single leg (TAICHI), dip on parallel bars (DIP) and a pull-up bar task (PULLUP). For the dip and pull-up workouts, we increase by 10 times the joint torque limits of the arms.[14] Additionally, we consider joint position and velocity limits in each scenario through a quadratic barrier. Both TAICHI, DIP and PULLUP tasks are divided into three phases; for the TAICHI task: manipulation, standing on one foot, and balancing; for the DIP and PULLUP bar task: grasping the bar, workout, and landing on ground. DIP and PULLUP problems have a horizon of $12\,\mathrm{s}$ with 400 nodes. The horizon of the whole-body manipulation and TAICHI problem are 3 and $6\,\mathrm{s}$ with 60 and 120 nodes, respectively. We define different Baumgarte gains for the hands and feet, i.e. $(0, 20)$ and $(0, 40)$, respectively. Furthermore, we do not include friction cone constraints in the grasping bar phase, i.e. for the hand grasping. We use the contact-placement cost functions for both: feet and hands. Indeed, behaviors such as dip / pull-up balancing, pull-up motion, or the leg-crossing strategy emerge from our optimization algorithm since we do not describe the center of mass and swing-contact motions. Finally, the regularization terms, dynamics and initialization strategy are the same that we use for the ANYmal cases. For the details about the ANYmal problems see above (Section A-C).

References

[1] J. T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed. USA: Cambridge University Press, 2009.

[2] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Rev.*, 2005.

[3] R. H. Byrd, J. Nocedal, and R. A. Waltz, "KNITRO: An integrated package for nonlinear optimization," in *Large Scale Nonlinear Optimization, 35–59, 2006*, 2006.

[4] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, 2006.

[5] 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.

[6] D. Pardo, L. Moller, M. Neunert, A. W. Winkler, and J. Buchli, "Evaluating Direct Transcription and Nonlinear Optimization Methods for Robot Motion Planning," *IEEE Robot. Automat. Lett. (RA-L)*, 2016.

[7] 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.

[8] 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.

[9] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2016.

[10] 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 Multi-Legged Locomotion via Mixed-Integer Convex Optimization," *IEEE Robot. Automat. Lett. (RA-L)*, 2017.

[11] A. W. Winkler, D. C. Bellicoso, M. Hutter, and J. Buchli, "Gait and Trajectory Optimization for Legged Systems through Phase-based End-Effector Parameterization," *IEEE Robot. Automat. Lett. (RA-L)*, 2018.

[12] W. Merkt, V. Ivan, and S. Vijayakumar, "Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2018.

[13] 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)*, 2020.

[14] 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.

[15] 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.

[16] 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.

[17] M. Neunert, M. Stauble, 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)*, 2018.

[18] 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.

[19] W. Li and E. Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," in *ICINCO*, 2004.

[20] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, 1966.

[21] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2018.

[22] 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.

[23] J. Albersmeyer and M. Diehl, "The Lifted Newton Method and Its Application in Optimization," *SIAM J. on Optimization*, 2010.

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

[25] J. Marti-Saumell, J. Sola, C. Mastalli, and A. Santamaria-Navarro, "Squash-Box Feasibility Driven Differential Dynamic Programming," in *IEEE/RSJ Int. Conf. Intell. Rob. Sys. (IROS)*, 2020.

[26] Y. Tassa, N. Mansard, and E. Todorov, "Control-Limited Differential Dynamic Programming," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2014.

[14]Talos' arms are not strong enough to support its own weight.

[27] K. Ohno, "A new approach to differential dynamic programming for discrete time systems," *IEEE Transactions on Automatic Control*, vol. 23, no. 1, pp. 37–47, 1978.

[28] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *IEEE Int. Conf. Rob. Autom. (ICRA)*, 2017.

[29] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guidance*, vol. 10, no. 4, 1987.

[30] G. Lantoine and R. Russell, "A Hybrid Differential Dynamic Programming Algorithm for Robust Low-Thrust Optimization," in *AIAA/AAS Astrodyn. Special. Conf. and Exhib.*, 2008.

[31] 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.

[32] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," in *Proc. Fast Motions in Biomechanics Robot.* Springer Berlin Heidelberg, 2006.

[33] I. Ross, *A primer on Pontryagin's principle in optimal control*, 2nd ed. San Francisco, USA: Collegiate Publishers, 2015.

[34] D. P. Bertsekas, "Projected newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization*, 1982.

[35] R. Fletcher, "A modified Marquardt subroutine for non-linear least squares," *J. Math. Sci.*, 1971.

[36] Y. Tassa, "Theory and Implementation of Biomimetic Motor Controllers," Ph.D. dissertation, Hebrew University of Jerusalem, 2011.

[37] C. Mastalli, R. Budhiraja, and N. Mansard, "Crocoddyl: a fast and flexible optimal control library for robot control under contact sequence," https://github.com/loco-3d/crocoddyl, 2019.

[38] 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.

[39] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An Interior Point Algorithm for Large-Scale Nonlinear Programming," *SIAM J. on Optimization*, 1999.

[40] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz, "An algorithm for nonlinear optimization using linear programming and equality constrained subproblems," *Math. Program.*, 2004.

[41] L. Hei, J. Nocedal, and R. A. Waltz, "A Numerical Study of Active-Set and Interior-Point Methods for Bound Constrained Optimization," in *Modeling, Simulation and Optimization of Complex Processes*, 2008.

[42] J. Leal, "CppADCodeGen," https://github.com/joaoleal/CppADCodeGen/, 2011–2020.

[43] B. Bell, "CppAD: A package for differentiation of C++ algorithms," in *ComputationalInfrastructure for Operations Research*, 2012. [Online]. Available: http://www.coin-or.org/CppAD/

[44] R. Featherstone, *Rigid Body Dynamics Algorithms.* Berlin, Heidelberg: Springer-Verlag, 2007.

[45] J. Carpentier and N. Mansard, "Analytical Derivatives of Rigid Body Dynamics Algorithms," in *Robotics: Science and Systems (RSS)*, 2018.

[46] 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.

[47] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, 1972.

Carlos Mastalli received the M.Sc. degree in mechatronics engineering from the Simón Bolívar University, Caracas, Venezuela, in 2013 and the Ph.D. degree in bio-engineering and robotics from the Istituto Italiano di Tecnologia, Genoa, Italy, in 2017.

He is currently a Research Associate in the University of Edinburgh with Alan Turing fellowship, Edinburgh, U.K. with S. Vijayakumar. From 2017 to 2019, he was a Postdoctoral Researcher in the Gepetto Team at LAAS-CNRS, Toulouse, France, working with N. Mansard and O. Stasse. Previously, he completed his Ph.D. on "Planning and Execution of Dynamic Whole-Body Locomotion on Challenging Terrain" under the supervision of I. Havoutis, C. Semini and D. G. Caldwell. He was Invited Researcher in the Eidgenössische Technische Hochschule Zürich, Zurich, Switzerland, with J. Buchli in 2016. His research combines the formalism of model-based approaches with the exploration of vast robot's data for legged robotics. He has contributions in optimal control, motion planning, whole-body control and machine learning for legged locomotion.

Wolfgang Merkt received the B.Eng.(Hns) degree in mechanical engineering with management and the M.Sc.(R) and Ph.D. degrees in robotics and autonomous systems from the University of Edinburgh, Edinburgh, U.K., in 2014, 2015 and 2019, respectively.

He is currently a Postdoctoral Researcher at the Oxford Robotics Institute, University of Oxford with I. Havoutis. During his Ph.D., he worked on trajectory optimization and warm starting optimal control for high-dimensional systems and humanoid robots under the supervision of S. Vijayakumar. His research interests include fast optimization-based methods for planning and control, loco-manipulation, and legged robots.

Josep Marti-Saumell received the B.Sc. degree in industrial engineering (majoring in mechanics) and the M.Sc. degree in automatic control and robotics from the Universitat Politècnica de Catalunya, Barcelona, Spain, in 2013 and 2018, respectively.

He is currently a Ph.D. candidate at the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. His current research interests include optimal control applied to mobile robotics, numerical optimization, and unmanned aerial manipulators.

Henrique Ferrolho received his M.Sc. degree in informatics and computing engineering from the University of Porto, Porto, Portugal in 2017.

He is currently pursuing a Ph.D. degree in robotics and autonomous systems at the University of Edinburgh under the supervision of S. Vijayakumar. His research interests include robust motion planning, and optimal control of agile robotic systems.

Joan Solà received the M.Sc. degree in telecommunication and electronics from the Universitat Politècnica de Catalunya, Barcelona, Spain, and the Ph.D. degree in robotics from the University of Toulouse, Toulouse, France, in 2007.

He is currently a Ramón y Cajal Researcher with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Barcelona, Spain. He has also worked in the industry in the renewable energies sector, and was involved in the construction of a manned submarine for depths up to 1200 m. He has contributed to monocular SLAM, especially in the undelayed initialization of landmarks, and is interested in state estimation for robots with particularly large dynamics and degrees of freedom, such as humanoids and aerial manipulators. His current projects turn around whole-body estimation and control, including multisensor fusion, localization and mapping, machine learning, and model predictive control.

Nicolas Mansard received the M.Sc. degree in computer science from the University of Grenoble, Grenoble, France, in 2003 and the Ph.D. degree in robotics from the University of Rennes, Rennes, France, in 2006.

He has been a CNRS Researcher since 2009. He was then Postdoctoral Researcher at Stanford University, Stanford, CA, USA with O. Khatib in 2007 and in JRL-Japan with A. Kheddar in 2008. He was Invited Researcher at the University of Washington with E. Todorov in 2014. He received the CNRS Bronze Medal in 2015 (one medal is awarded in France in automatic/robotic/signal-processing every year). His main research interests include the motion generation, planning and control of complex robots, with a special regard in humanoid robotics. His expertise covers sensor-based (vision and force) control, numerical mathematics for control, bipedal locomotion and locomotion planning. He published more than 70 papers in international journals and conferences and supervised 10 Ph.D. theses.

Dr. Mansard is currently an Associate Editor of the IEEE TRANSACTIONS ON 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.