A Direct-Indirect Hybridization Approach to Control-Limited DDP

Carlos Mastalli* Josep Marti-Saumell† Wolfgang Merkt*
Joan Solà† Nicolas Mansard‡ Sethu Vijayakumar*
*School of Informatics, University of Edinburgh, Edinburgh, UK
†Institut de Robòtica i Informàtica Industrial, Universitat Politècnica de Catalunya, Barcelona, Spain
‡Gepetto Team, LAAS-CNRS, Toulouse, France
Email: carlos.mastalli@ed.ac.uk.

***Abstract*—Optimal control is a widely used tool for synthesizing motions and controls for user-defined tasks under physical constraints. A common approach is to formulate it using direct multiple-shooting and then to use off-the-shelf nonlinear programming solvers that can easily handle arbitrary constraints on the controls and states. However, these methods are not fast enough for many robotics applications such as real-time humanoid motor control. Exploiting the sparse structure of optimal control problem, such as in Differential Dynamic Programming (DDP), has proven to significantly boost the computational efficiency, and recent works have been focused on handling arbitrary constraints. Despite that, DDP has been associated with poor numerical convergence, particularly when considering long time horizons. One of the main reasons is due to system instabilities and poor warm-starting (only controls). This paper presents *control-limited Feasibility-driven DDP* (Box-FDDP), a solver that incorporates a direct-indirect hybridization of the control-limited DDP algorithm. Concretely, the forward and backward passes handle feasibility and control limits. We showcase the impact and importance of our method on a set of challenging optimal control problems against the Box-DDP and squashing-function approach.**

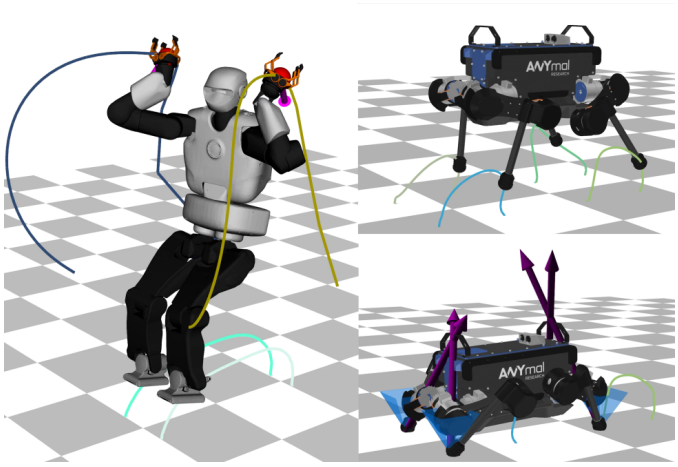

Fig. 1. Box-FDDP: a solver that incorporates a direct-indirect hybridization of the control-limited DDP algorithm. Challenging maneuvers computed by Box-FDDP: monkey bar and jumping tasks. The lines describe the trajectory performed by the hands and the feet.

I. Introduction

A. Motivation

Optimal control is a powerful tool to synthesize motions and controls through task goals (cost / optimality) and constraints (e.g., system dynamics, interaction constraints, etc.). We can formulate it through *direct methods* [5], which discretize over both state and controls as optimization variables, and then use general-purpose Nonlinear Programming (NLP) solvers such as SNOPT [15], KNITRO [7], and IPOPT [29]. However, the main disadvantage of this approach is that it requires very large matrix factorizations, which limits its application domain to control on reduced models (e.g. [30, 25, 10]) or motion planning (e.g. [9, 20, 1, 31]).

Recent results on fast nonlinear Model Predictive Control (nMPC) based on DDP (e.g. [26, 18, 23, 12]) have once again attracted attention to *indirect methods* which only discretize over the controls, in particular with its Gauss-Newton (GN) approximation called iterative Linear-Quadratic Regulator (iLQR) [19]. These methods impose and exploit a sparse structure of the problem by applying "Bellman's principle of optimality" and successively solving the smaller sub-problems. This leads to fast and cheap computation due to very small matrix factorizations and effective data cache accesses. On the contrary, the main limitation compared with direct methods is the ability to efficiently encode hard constraints.

B. Related work

There are important recent achievements in both *direct-indirect* hybridization [14, 21] and handling input limits [27], as well as nonlinear constraints [32, 17], that are rooted in dynamic programming. For instance, Giftthaler et al. [14] introduced a *lifted*[1] version of the Riccati equations that allows us to warm-start both state and control trajectories. In turn, Mastalli et al. [21] proposed a modification of the forward pass that numerically matches the gap contraction expected by a *direct multiple-shooting* method with only equality constraints. Unfortunately, both methods do not handle inequality constraints such as control limits. However, their hybrid approach makes these works numerically more robust to poor initialization as well as initialization from state trajectories. With hybrid approach, we refer to artificially include defect

[1]This name is coined by [2], and we refer to *gaps* or *defects* produced between multiple shooting nodes.

constraints (i.e. gaps on the dynamics) and the state trajectory as decision variables.

Tassa et al. [27] focused on handling control limits during the computation of the backward pass in a DDP-like method (Box-DDP). They proposed to include control bounds during the minimization of the control Hamiltonian (i.e. the $\mathbf{Q}$ terms). Later, Xie et al. [32] included general inequality constraints into Tassa's method (CDDP). Despite that, Xie's method sacrifices the computational effort by including a second Quadratic Programming (QP) program in the forward pass, even though it remains faster than solving the same problem using a direct formulation (with SNOPT). To reduce the computational cost, Howell et al. [17] formulate the inequality constraints using an Augmented Lagrangian method (ALTRO), which also uses an active-set projection for solution polishing. In this work, we bridge the gap between direct-indirect hybridization with control-limits inequalities. Other inequalities, e.g. on the state, are handled through quadratic penalization as explained below.

C. Contribution

In this paper, we propose enhancements to the Box-DDP algorithm [27]. These modifications make our method more robust (a) to face the *feasibility* problem, and (b) to discover good solutions despite poor initial guesses. Our algorithm is called control-limited Feasibility-driven Differential Dynamic Programming (FDDP) (in short Box-FDDP). It comprises two modes: *feasibility-driven* and *control-bounded* modes. Box-FDDP combines a hybridization of the multiple-shooting with DDP to leverage feasibility under control limits. In our results, we highlight the impact and importance of these changes over a wide range of different optimal control problems: from double pendulum to humanoid locomotion (Fig. 1).

The rest of the paper is organized as follows. In Section II, we quickly describe the optimal control formulation and then recall the DDP algorithm with control limits. Section III describes our algorithm called Box-FDDP. Results that support the impact and importance of our proposed changes are provided for several optimal control problems in Section IV, and Section V summarizes the work conclusions.

II. Preliminaries

In this section, we first introduce our formulation of the multi-phase optimal control problem (Section II-A). We use holonomic constraints, similarly to [16, 6], together with friction-cone inequalities to model contacts. Our main difference is that we efficiently compute the derivatives of the contact forces as explained in Section II-A1. Later, we give a short introduction to the Box-DDP algorithm (Section II-B), necessary for our proposed algorithm called Box-FDDP; for a complete description see [22, 26].

A. Optimal control formulation

Given a predefined contact sequence and timings, we formulate a hybrid optimal control problem that exploits the sparsity of the contact dynamics and its derivatives[2]:

$$
\begin{aligned}
\min_{\mathbf{x}_s,\mathbf{u}_s}\ & l_N(\mathbf{x}_N) + \sum_{k=0}^{N-1} \int_{t_k}^{t_k+\Delta t_k} l_k(\mathbf{x}_k,\mathbf{u}_k,\boldsymbol{\lambda}_k)dt \\
\text{s.t.}\quad & \mathbf{q}_{k+1} = \mathbf{q}_k \oplus \int_{t_k}^{t_k+\Delta t_k} \mathbf{v}_k\, dt, && \text{(integrator)} \\
& \mathbf{v}_{k+1} = \mathbf{v}_k + \int_{t_k}^{t_k+\Delta t_k} \dot{\mathbf{v}}_k\, dt, \\
& \begin{bmatrix} \dot{\mathbf{v}}_k \\ -\boldsymbol{\lambda}_k \end{bmatrix} = \begin{bmatrix} \mathbf{M} & \mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{\tau}_b \\ -\mathbf{a}_0 \end{bmatrix}, && \text{(contact dynamics)} \\
& \mathbf{R}\boldsymbol{\lambda}_{\mathcal{C}(k)} \leq \mathbf{r}, && \text{(friction-cone)} \\
& \log\left(\mathbf{p}_{\mathcal{G}(k)}(\mathbf{q}_k)^{-1} \mathbf{o} \mathbf{M}_{\mathbf{f}_{\mathcal{G}(k)}}\right) = \mathbf{0}, && \text{(contact placement)} \\
& \underline{\mathbf{x}} \leq \mathbf{x}_k \leq \bar{\mathbf{x}}, && \text{(state bounds)} \\
& \underline{\mathbf{u}} \leq \mathbf{u}_k \leq \bar{\mathbf{u}}, && \text{(control bounds)}
\end{aligned}
\tag{1}
$$

where the state $\mathbf{x} = (\mathbf{q}, \mathbf{v}) \in X$ lies in a differential manifold (with n_x dimension) formed by the configuration point $\mathbf{q}$ and its tangent vector $\mathbf{v}$, the control $\mathbf{u} \in \mathbb{R}^{n_u}$ is formed by the input torque commands, $\mathcal{C}(k)$ and $\mathcal{G}(k)$ define respectively the set of active and gain contacts given the time index k, $\mathbf{M} \in \mathbb{R}^{n_x \times n_x}$ is the joint-space inertia matrix, $\mathbf{J}_c$ is the contact Jacobian, $\boldsymbol{\lambda} \in \mathbb{R}^{n_f}$ is the contact force, $\mathbf{a}_0 \in \mathbb{R}^{n_f}$ is the desired acceleration in the constraint space, $\boldsymbol{\tau}_b$ is the force-bias vector, and $(\mathbf{R}, \mathbf{r})$ describe the linearized friction cone. Note that we could alternatively use impulse dynamics during the contact transition phases as described in [21].

Regarding the cost functions $l_k(\cdot)$, we use a naïve reference of the center of mass and swing motions to drive the solution towards a desired behavior. To improve numerical stability, especially important for redundant tasks, we apply regularization on the contact dynamics, centroidal momentum, state, and control. For efficiency purposes, we use the GN approximation to compute the Hessians of the cost function (e.g. $\mathbf{l_{xx}} = \mathbf{l_x}^T \mathbf{l_x}$).

The contact dynamics [28] are defined by (a) the stack of contact Jacobian expressed in the local frame $\mathbf{J}_c$ (full-rank matrix) and defined by $\mathcal{C}(k)$ indices, (b) the desired acceleration in the constraint space $\mathbf{a}_0 \in \mathbb{R}^{n_f}$ (with Baumgarte stabilization [3]), (c) the joint-space inertia matrix $\mathbf{M} \in \mathbb{R}^{n_x \times n_x}$, and (d) the force-bias vector $\boldsymbol{\tau}_b = \mathbf{S}\mathbf{u} - \mathbf{b} \in \mathbb{R}^{n_x}$ that accounts for control $\mathbf{u}$, the Coriolis and gravitational effect $\mathbf{b}$, and $\mathbf{S}$ selects the actuated joint coordinates. We regularize the Schur complement during the LDU decomposition needed for the matrix inversion of the contact dynamics. Additionally, during the contact-gain phases [13], we substitute the contact dynamics by impulses of the form:

$$
\begin{bmatrix} \mathbf{M} & \mathbf{J}_c^\top \\ \mathbf{J}_c & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}^+ \\ -\boldsymbol{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{M}\mathbf{v}^- \\ -e\mathbf{J}_c\mathbf{v}^- \end{bmatrix}, \tag{2}
$$

where $\boldsymbol{\Lambda}$ is the contact impulse and, $\mathbf{v}^-$ and $\mathbf{v}^+$ are the discontinuous changes in the generalized velocity (i.e., velocity

[2]We exploit the sparse structure of $\mathbf{M}$ and $\mathbf{J}_c$ during the matrix-matrix multiplications needed for the factorization.

before and after impact, respectively), and $e \in [0,1]$ is the restitution coefficient that considers compression / expansion.

The friction-cone, contact placement, and state-bounds constraints are enforced through quadratic penalization, which in practice has good numerical convergence compared with the log-barrier method. On the contrary, control bounds are handled as explained in Section III. The linearized friction-cone model $(\mathbf{R}, \mathbf{r})$ is computed from the surface normal $\hat{n}$, friction coefficient μ, and minimum / maximum normal forces $(\underline{\boldsymbol{\lambda}}, \bar{\boldsymbol{\lambda}})$. As the contact placement constraint lies on a $\mathbb{SE}(3)$, we explicitly express it in its tangent space using the $\log(\cdot)$ operator.

1) Contact forces and their derivatives: As we express contact forces as accelerations in the contact constraint subspace, we analytically compute the derivatives of the contact-forces for imposing the friction-cone constraints. If we apply the chain rule, the first-order Taylor approximation of the contact forces is defined as:

$$\delta\boldsymbol{\lambda} = \overbrace{\begin{bmatrix}\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{\tau}} & \frac{\partial\boldsymbol{\lambda}}{\partial\mathbf{a}_0}\end{bmatrix}\begin{bmatrix}\frac{\partial\boldsymbol{\tau}}{\partial\mathbf{x}} \\ \frac{\partial\mathbf{a}_0}{\partial\mathbf{x}}\end{bmatrix}}^{\boldsymbol{\lambda}_\mathbf{x}}\delta\mathbf{x} + \overbrace{\begin{bmatrix}\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{\tau}} & \frac{\partial\boldsymbol{\lambda}}{\partial\mathbf{a}_0}\end{bmatrix}\begin{bmatrix}\frac{\partial\boldsymbol{\tau}}{\partial\mathbf{u}} \\ \frac{\partial\mathbf{a}_0}{\partial\mathbf{u}}\end{bmatrix}}^{\boldsymbol{\lambda}_\mathbf{u}}\delta\mathbf{u}, \quad (3)$$

where $\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{\tau}} = -\widehat{\mathbf{M}}^{-1}\mathbf{J}_c\mathbf{M}^{-1}$ and $\frac{\partial\boldsymbol{\lambda}}{\partial\mathbf{a}_0} = -\widehat{\mathbf{M}}^{-1}$ are the contact-force Jacobians with respect to inverse dynamics and constrained-acceleration kinematics functions, respectively, $\frac{\partial\boldsymbol{\tau}}{\partial\mathbf{x}}$, $\frac{\partial\boldsymbol{\tau}}{\partial\mathbf{u}}$ are the derivatives of Recursive Newton-Euler Algorithm (RNEA), and $\frac{\partial\mathbf{a}_0}{\partial\mathbf{x}}$, $\frac{\partial\mathbf{a}_0}{\partial\mathbf{u}}$ are the kinematics derivatives of the frame acceleration computed as in [8]. A similar procedure is used to compute the derivatives of the impulse dynamics described in Eq. (2).

B. Differential dynamic programming with control-limits

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

$$\begin{aligned} V_k(\delta\mathbf{x}_k) = \min_{\delta\mathbf{u}_k} & \; l_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) + V_{k+1}(\mathbf{f}_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k)), \\ \text{s.t.} & \quad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}}\ , \end{aligned} \quad (4)$$

which breaks the constrained Optimal Control (OC) problem into a sequence of simpler sub-problems; $\underline{\mathbf{u}}$, $\bar{\mathbf{u}}$ are the lower and upper bounds of the control, respectively. Then, a local search direction is computed through a Linear Quadratic (LQ) approximation of the Value function:

$$\delta\mathbf{u}_k^*(\delta\mathbf{x}_k) = \quad (5)$$

$$\arg\min_{\delta\mathbf{u}_k} \frac{1}{2}\overbrace{\begin{bmatrix}1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix}^T\begin{bmatrix}0 & \mathbf{Q}_{\mathbf{x}_k}^T & \mathbf{Q}_{\mathbf{u}_k}^T \\ \mathbf{Q}_{\mathbf{x}_k} & \mathbf{Q}_{\mathbf{x}\mathbf{x}_k} & \mathbf{Q}_{\mathbf{x}\mathbf{u}_k} \\ \mathbf{Q}_{\mathbf{u}_k} & \mathbf{Q}_{\mathbf{x}\mathbf{u}_k}^T & \mathbf{Q}_{\mathbf{u}\mathbf{u}_k}\end{bmatrix}\begin{bmatrix}1 \\ \delta\mathbf{x}_k \\ \delta\mathbf{u}_k\end{bmatrix}}^{\mathbf{H}(\delta\mathbf{x}_k, \delta\mathbf{u}_k, \bar{V}_k, k)},$$

$$\text{s.t.} \quad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}}, \quad (6)$$

where the $\mathbf{Q}_k$ terms represent the LQ approximation of the Hamiltonian function $\mathbf{H}(\cdot)$, and the derivatives of Value function $\bar{V}_k = (V_{\mathbf{x}_k}, V_{\mathbf{x}\mathbf{x}_k})$ take the role of the costate variables. From the derivatives of the cost and dynamics functions $l_k(\cdot)$, $\mathbf{f}_k(\cdot)$, the Hamiltonian is calculated around a guess $(\mathbf{x}_k^i, \mathbf{u}_k^i)$ at each i-th iteration [22]. Solving Eq. (5) provides the feed-forward term $\mathbf{k}_k$ and the feedback gain $\mathbf{K}_k$ at each discretization point k.

1) Control-bounded direction: Due to the mathematical simplicity of the control bounds, a subspace minimization approach allows the active set to change rapidly [24]. The subspace is defined by the search direction projected onto the feasible box. To adopt this strategy into DDP algorithm, Tassa et al. [27] proposed to break the problem into feed-forward and feedback sub-problems, where the feed-forward problem is defined as

$$\begin{aligned} \mathbf{k}_k = \arg\min_{\delta\mathbf{u}_k} & \; \frac{1}{2}\delta\mathbf{u}_k^T\mathbf{Q}_{\mathbf{u}\mathbf{u}_k}\delta\mathbf{u}_k + \mathbf{Q}_{\mathbf{u}_k}^T\delta\mathbf{u}_k, \\ \text{s.t.} & \quad \underline{\mathbf{u}} \leq \mathbf{u}_k + \delta\mathbf{u}_k \leq \bar{\mathbf{u}}, \end{aligned} \quad (7)$$

and it is solved by iteratively identifying the active set and then moving along the free subspace of the Newton step (i.e. Projected-Newton QP [4]). Instead, the feedback gain is computed along the free subspace of the Hessian, i.e.

$$\mathbf{K}_k = -\mathbf{Q}_{\mathbf{u}\mathbf{u},f_k}^{-1}\mathbf{Q}_{\mathbf{u}\mathbf{x}_k}, \quad (8)$$

where $\mathbf{Q}_{\mathbf{u}\mathbf{u},f_k}^{-1}$ is the control Hessian of the free subspace, and it is computed from feed-forward sub-problem. Note that the Box-QP computes the Newton direction along the free subspace.

With this feedback gain, the changes in the nominal trajectory are projected onto the feasible box. Additionally, the Box-QP algorithm requires a feasible warm-start $\delta\mathbf{u}_k^0$, and if it has the same active set, then the solution is calculated within a single iteration[3].

III. Control-limited FDDP

The Box-FDDP comprises two modes: *feasibility-driven* and *control-bounded* modes, that might be chosen in a given iteration (Algorithm 1). The feasibility-driven mode uses a DDP hybridization of the multiple-shooting formulation to compute the search direction and step length (lines 7 and 15). Instead, the control-bounded mode projects the search direction onto the feasible control region whenever the dynamics constraint is feasible (line 10). Additionally, the applied control is always projected onto its feasible box (line 13), causing dynamic-infeasible iterations to reach the control box. Technical descriptions of both modes are elaborated in Sections III-A and III-B.

A. Search direction of Box-FDDP

In direct multiple-shooting, the nonlinearities of the dynamics are distributed over the entire horizon, instead of being accumulated as in single shooting [11]. In DDP, the feedback gain helps to distribute the dynamics nonlinearities as well. However, it does not resemble the Hamilton-Jacobi-Bellman (HJB) equation applied to a direct multiple-shooting formulation as described below.

[3]The computational cost of a single iteration is similar to performing a Cholesky decomposition.

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. (9)
4  for k ← N − 1 to 0 do
5      update the feasibility-driven Hamiltonian, Eq. (12)
6      if infeasible iterate then
7          compute feasibility-driven direction, Eq. (14)
8      else
9          clamp Box-QP warm-start, Eq. (15)
10         compute control-bounded direction, Eq. (7)-(8)
11 for α ∈ {1, 1/2, ··· , 1/2^n} do
12     for k ← 0 to N do
13         project control onto the feasible box, Eq. (16)
14         if infeasible iterate or α ≠ 1 then
15             update the gaps, Eq. (17)
16         else
17             close the gaps,
                 f_k = 0  ∀k ∈ {0, ··· , N − 1}
18         perform step, Eq. (19)
19     compute the expected improvement, Eq. (20)
20     if success step then
21         break
```

1) Computing the gaps: Given a current iterate $(\mathbf{x}_s, \mathbf{u}_s)$, we compute the gaps by performing a nonlinear rollout, i.e.

$$\bar{\mathbf{f}}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}, \tag{9}$$

where $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is the rollout state at interval $k+1$, and $\mathbf{x}_{k+1}$ is the next shooting state.

In the standard Box-DDP, an initial forward pass is performed in order to close the gaps. Instead, our Box-FDDP computes the gaps once at each iteration (line 3), and then uses them to find the search direction (Section III-A3) and to compute the expected improvement (Section III-B4).

2) Hamiltonian of direct multiple-shooting formulation: Without loss of generality, we use the GN approximation [19] to write the Hamiltonian function as

$$\begin{aligned}\mathbf{H}(\cdot) = &\frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_{k+1}\end{bmatrix}^T\begin{bmatrix}0 & V_{\mathbf{x}_{k+1}}^T\\ V_{\mathbf{x}_{k+1}} & V_{\mathbf{xx}_{k+1}}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_{k+1}\end{bmatrix}\\ &+\frac{1}{2}\begin{bmatrix}1\\ \delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix}^T\begin{bmatrix}0 & \mathbf{l}_{\mathbf{x}_k}^T & \mathbf{l}_{\mathbf{u}_k}^T\\ \mathbf{l}_{\mathbf{x}_k} & \mathbf{l}_{\mathbf{xx}_k} & \mathbf{l}_{\mathbf{xu}_k}\\ \mathbf{l}_{\mathbf{u}_k} & \mathbf{l}_{\mathbf{xu}_k}^T & \mathbf{l}_{\mathbf{uu}_k}\end{bmatrix}\begin{bmatrix}1\\ \delta\mathbf{x}_k\\ \delta\mathbf{u}_k\end{bmatrix},\end{aligned} \tag{10}$$

where $\mathbf{l}_\mathbf{x}, \mathbf{l}_\mathbf{u}$ and $\mathbf{l}_\mathbf{xx}, \mathbf{l}_\mathbf{xu}, \mathbf{l}_\mathbf{uu}$ are 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 $\mathbf{f}_\mathbf{x}, \mathbf{f}_\mathbf{u}$ are its Jacobians. However, in a direct multiple-shooting setting, we have a drift in the linearized dynamics

$$\delta\mathbf{x}_{k+1} = \mathbf{f}_{\mathbf{x}k}\delta\mathbf{x}_k + \mathbf{f}_{\mathbf{u}k}\delta\mathbf{u}_k + \bar{\mathbf{f}}_{k+1} \tag{11}$$

due to the gaps $\bar{\mathbf{f}}_{k+1}$ produced between multiple-shoots. Then, according to the Pontryagin's Maximum Principle (PMP), the Riccati recursion needs to be adapted as follows:

$$\begin{aligned}\mathbf{Q}_{\mathbf{x}_k} &= \mathbf{l}_{\mathbf{x}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{x}_{k+1}}^+,\\ \mathbf{Q}_{\mathbf{u}_k} &= \mathbf{l}_{\mathbf{u}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{x}_{k+1}}^+,\\ \mathbf{Q}_{\mathbf{xx}_k} &= \mathbf{l}_{\mathbf{xx}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{x}_k},\\ \mathbf{Q}_{\mathbf{xu}_k} &= \mathbf{l}_{\mathbf{xu}_k} + \mathbf{f}_{\mathbf{x}_k}^T V_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{u}_k},\\ \mathbf{Q}_{\mathbf{uu}_k} &= \mathbf{l}_{\mathbf{uu}_k} + \mathbf{f}_{\mathbf{u}_k}^T V_{\mathbf{xx}_{k+1}}\mathbf{f}_{\mathbf{u}_k},\end{aligned} \tag{12}$$

where

$$V_{\mathbf{x}_{k+1}}^+ = V_{\mathbf{x}_{k+1}} + V_{\mathbf{xx}_{k+1}}\bar{\mathbf{f}}_{k+1} \tag{13}$$

is the Jacobian of the Value function after the deflection produced by $\bar{\mathbf{f}}_{k+1}$, and the Hessian of the Value function remains unchanged. Indeed, this is possible since DDP approximates the Value function to a LQ model. Note that a similar derivation is proposed by [14].

3) Feasibility-driven direction: During *dynamic-infeasible* iterates (line 7), we compute a *control-free* direction[4]:

$$\begin{aligned}\mathbf{k}_k &= -\mathbf{Q}_{\mathbf{uu}_k}^{-1}\mathbf{Q}_{\mathbf{u}_k},\\ \mathbf{K}_k &= -\mathbf{Q}_{\mathbf{uu}_k}^{-1}\mathbf{Q}_{\mathbf{ux}_k}.\end{aligned} \tag{14}$$

The reason is due to the fact that we cannot quantify the effect of the gaps on the control bounds, which are needed to solve the feed-forward sub-problem Eq. (7). Note that our approach is equivalent to opening the control bounds during dynamic-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, in case of a previous infeasible iteration, $\mathbf{k}_k$ might fall outside the feasible box (i.e. $\bar{\mathbf{u}} - \delta\mathbf{u}_k \leq \mathbf{k}_k \leq \underline{\mathbf{u}} - \delta\mathbf{u}_k$). This is in contrast to the standard Box-DDP, in which a feasible warm-start $\mathbf{u}_s^0$ needs to be provided, and then, the iterates always remain feasible.

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

$$[\![\mathbf{k}_k]\!]_{\mathbf{u}-\delta\mathbf{u}_\mathbf{k}} = \min\left(\max\left(\mathbf{k}_k, \bar{\mathbf{u}} - \delta\mathbf{u}_k\right), \underline{\mathbf{u}} - \delta\mathbf{u}_k\right), \tag{15}$$

where $\bar{\mathbf{u}} - \delta\mathbf{u}_k$ and $\underline{\mathbf{u}} - \delta\mathbf{u}_k$ are the upper and lower bounds of the feed-forward sub-problem, Eq. (7), respectively.

5) Regularization: Each time that the computation of the feed-forward sub-problem fails, Eq. (7), we increment the regularization over $\mathbf{Q}_\mathbf{uu}$ and re-start the computation of the direction. On the other hand, each time that the algorithm accepts a big step[5], then we decrease the regularization. With this, we provide major robustness to the algorithm since it moves from Newton direction to steepest-descent, or vice versa.

[4]In this work, with *control-free* direction, we also refer to feasibility-driven direction, i.e., the direction ignoring the control constraints.

[5]Steps with $\alpha \geq \alpha_0$, where α_0 is an user-defined threshold.

B. Step-length of Box-FDDP

As far as we know, Tassa et al. [26] proposed only to modify the search direction using a Box-QP. However, it is important to pay attention to the rollout as well, during which we find a step length that minimizes the cost [24]. A similar motivation can be found in methods such as [32, 17], where a standard line-search procedure is used around a local model.

1) Projecting the rollout towards the feasible box: We propose to project the control onto the feasible box in the nonlinear rollout (line 13), i.e.

$$\hat{\mathbf{u}}_k \leftarrow \min\left(\max\left(\hat{\mathbf{u}}_k, \bar{\mathbf{u}}\right), \underline{\mathbf{u}}\right), \tag{16}$$

where $\hat{\mathbf{u}}_k$ is the control policy computed from the search direction; for more details see Section III-B3. Our method does not require to solve another QP problem [32] or to project the linear search direction given the gaps on the dynamics [17].

2) Updating the gaps: In related work [21] is analyzed the behavior of the gaps during the numerical optimization, i.e. by iteratively solving the Karush-Kuhn-Tucker (KKT) problem of a direct multiple-shooting algorithm. Their conclusion is that the gaps will be either partially closed by a factor of

$$\bar{\mathbf{f}}_k \leftarrow (1-\alpha)\bar{\mathbf{f}}_k, \tag{17}$$

where α is the accepted step-length found by the line-search procedure (line 11-21), or completely closed in case of a full-step ($\alpha = 1$).

3) Nonlinear step: With a nonlinear rollout[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 solvers. Therefore, the prediction of gaps after applying an α-step are:

$$\begin{aligned} \bar{\mathbf{f}}^{i+1}_{k+1} &= \bar{\mathbf{f}}^{i}_{k+1} - \alpha(\delta\mathbf{x}_{k+1} - \mathbf{f}_{\mathbf{x}k}\delta\mathbf{x}_k - \mathbf{f}_{\mathbf{u}k}\delta\mathbf{u}_k) \\ &= (1-\alpha)(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}), \end{aligned} \tag{18}$$

and if we keep the gap-contraction rate of Eq. (17), then we obtain

$$\begin{aligned} \hat{\mathbf{x}}_k &= \mathbf{f}(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_{k-1}) - (1-\alpha)\bar{\mathbf{f}}_{k-1}, \\ \hat{\mathbf{u}}_k &= \mathbf{u}_k + \alpha\mathbf{k}_k + \mathbf{K}_k(\hat{\mathbf{x}}_k - \mathbf{x}_k), \end{aligned} \tag{19}$$

where $\mathbf{k}_k$ and $\mathbf{K}_k$ are the feed-forward term and feedback gains computed by Eq. (14) or Eq. (7)-(8), and the initial condition of the rollout is defined as $\hat{\mathbf{x}}_0 = \tilde{\mathbf{x}}_0 - (1-\alpha)\bar{\mathbf{f}}_0$. This is in contrast to the standard Box-DDP, in which the gaps are always closed.

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

$$\Delta J(\alpha) = \Delta_1\alpha + \frac{1}{2}\Delta_2\alpha^2, \tag{20}$$

with

$$\begin{aligned} \Delta_1 &= \sum_{k=0}^{N} \mathbf{k}_k^\top \mathbf{Q}_{\mathbf{u}_k} + \bar{\mathbf{f}}_k^\top (V_{\mathbf{x}_k} - V_{\mathbf{x}\mathbf{x}_k}\hat{\mathbf{x}}_k), \\ \Delta_2 &= \sum_{k=0}^{N} \mathbf{k}_k^\top \mathbf{Q}_{\mathbf{u}\mathbf{u}_k}\mathbf{k}_k + \bar{\mathbf{f}}_k^\top (2V_{\mathbf{x}\mathbf{x}_k}\mathbf{x}_k - V_{\mathbf{x}\mathbf{x}_k}\bar{\mathbf{f}}_k). \end{aligned} \tag{21}$$

Note that J is the total of cost of a given state-control trajectory $(\mathbf{x}_s, \mathbf{u}_s)$.

We obtain this expression by computing the cost from a linear rollout of the current control policy as described in Eq. (19). Finally, we also accept ascend directions since we use the Goldstein condition to check for the trial step.

IV. Results

Box-FDDP outperforms Box-DDP on a wide number of optimal control problems which are described in Section IV-A. In Section IV-B, we provide a comparison that shows the advantages of the proposed modifications. Later, we analyze the gap contraction and how it is connected with the dynamic nonlinearities (Section IV-B2). Finally, we show that the early control saturation of the Box-DDP has disadvantages (Section IV-C).

A. Optimal control problems

We compare the performance of our solver against Box-DDP [27] for a range of different OC problems: an under-actuated double pendulum, a quadcopter navigating through a narrow passage and looping, various gaits in legged locomotion, aggressive jumps and unstable hopping, and whole-body manipulation and balance. All the studied cases highlight the benefits of our proposed method: Box-FDDP. To make a fair comparison, we use the same initial regularization value (10^{-9}) and stopping criteria. Fig. 2 shows snapshots of motion computed for some of these problems, for more details see the accompanying video[7].

1) 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 $5\,\mathrm{N}$, largely insufficient for a static execution of the trajectory). The time horizon is $1\,\mathrm{s}$ with 100 nodes.

2) Quadcopter: We consider three tasks for the IRIS quadcopter: reaching goal (quad), looping maneuver (loop), and traversing a narrow passage (narrow). We use different way-points to describe the task, where each way-point specifies the desired pose and velocity. The way-points are described through cost functions in the robot placement and velocity. The vehicle pose is described as $\mathbb{SE}(3)$ 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. The solution is computed from a cold-start of the solver.

[6] In this work, *rollout* is sometimes referred as nonlinear step.

[7] https://www.dropbox.com/s/oyrfqijbrajgj0i

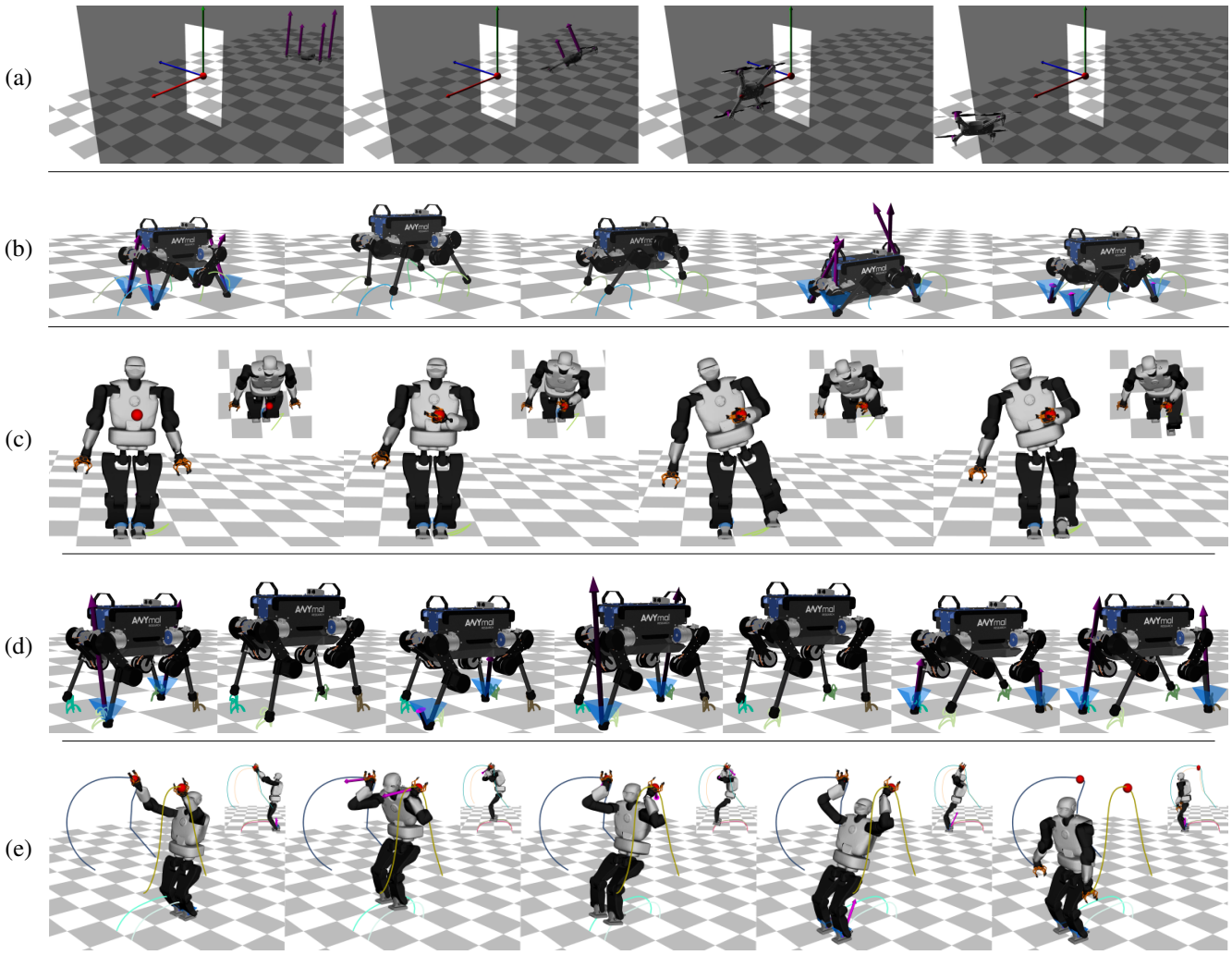

Fig. 2. Snapshots of generated robot maneuvers using Box-FDDP. (a) traversing a narrow passage with a quadcopter (quad); (b) aggressive jumping of 30 cm that reaches ANYmal limits (jump); (c) Talos balancing on a single leg (taichi); (d) ANYmal hopping with two legs (hop); (e) Talos climbing in a monkey bar.

3) 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. The joint velocity limits make it particularly hard to solve the jumping task (jump). Finally, the unstable hopping task (hop) is described with a long horizon: $5.8\,\mathrm{s}$ with 580 nodes. It includes 10 hops in total with a phase that switches the legs. We warm-start the solver with the default posture and quasi-static torques[8].

4) Whole-body manipulation and balance: We consider three problems for the Talos humanoid robot: whole-body manipulation (man), hand control while balancing in single leg (taichi), and a monkey bar task (bar). For the monkey bar task, we increase by 10 times the joint torque limits of the arms[9]. Additionally, we consider joint position limits in each scenario. Both taichi and monkey bar tasks are divided in three phases; for the taichi task: manipulation, standing on one foot, and balancing; for the monkey bar task: grasping the bar, climbing up, and landing on ground. Note that we do not include friction cone constraints in the grasping bar phase.

[8]We use the reference posture with zero velocity to compute the quasi-static torques, i.e. the torques due to the effect of gravity.

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

B. Advantages of the feasibility mode

To understand the benefits of the feasibility-mode, we analyze the resulting total cost and number of iterations for both: Box-FDDP and Box-DDP. As shown in Fig. 3, Box-FDDP's solutions (*-feas) have lower total cost and are computed with fewer iterations when compared to Box-DDP. We summarize the results of each benchmark problem in Table I.

1) Greater globalization strategy: The feasibility-driven mode becomes crucial to solve the double pendulum (pend), monkey bar task (bar), aggressive jumping (jump), and unstable hopping (hop) problems, in which the Box-DDP fails to find a solution (Table I). This mode helps to find a feasible sequence of controls despite the poor initialization warm-start. Indeed, infeasible iterations can be seen as a globalization

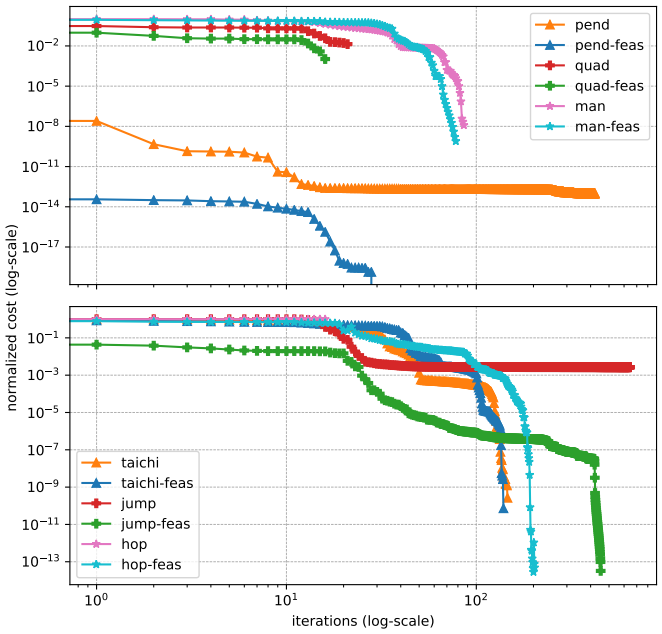

Fig. 3. Cost and convergence comparison for different optimal control problems. The Box-FDDP outperforms the Box-DDP in all the cases: (top) double pendulum (pend), quadcopter navigation (quad), and whole-body manipulation (man); and (bottom) whole-body balance (taichi), quadrupedal jumping (jump), quadrupedal hopping (hop). Box-FDDP (*-feas) solves the problem with fewer iterations and lower cost than Box-DDP. Furthermore, Box-DDP fails to solve the hardest problems: i.e. double pendulum, quadrupedal jumping, and hopping. Our algorithm shows better globalization strategy, i.e. it is less sensitive to poor initialization compared with Box-DDP.

TABLE I
NUMBER OF ITERATIONS, TOTAL COST, AND SOLUTION SUCCESS RATE.

	Box-DDP			Box-FDDP (feas)		
Problems	Iter.	Cost	Sol.	Iter.	Cost	Sol.
pend	424	0.223	✗	31	0.0273	✓
quad-goal	23	0.0764	✓	18	0.0072	✓
quad-loop	133	6.7211	✓	56	0.6444	✓
quad-narrow	70	1.9492	✓	35	0.4577	✓
man	88	4.6193	✓	80	4.6193	✓
taichi	148	6.8184	✓	141	6.8184	✓
jump	646	7.21×10^4	✗	454	1.81×10^4	✓
hop	18	1.13×10^6	✗	205	2.23×10^4	✓
bar	27	927.7	✗	358	23.316	✓

✓ solver finds a solution, ✗ solver does not find a solution.

strategy that ensure convergence from remote initial points as they balance objective and feasibility. We encountered that this trade-off could also improve the solution. For instance, early clamping of control commands (produced by Box-DDP) generates unnecessary loops during the quadcopter navigation.

2) Gap contraction and nonlinearities: Fig. 4 shows the gap contraction for each benchmark problem. We observe that the gap contraction rate is highly influenced by the nonlinearities of the system's dynamics. When compared to the dynamics, the nonlinearities of the task have a smaller effect (e.g. jump vs hop). Note that the gap contraction speed follows the order: humanoid, quadruped, double pendulum, and quadcopter.

Propagation errors due to the dynamics linearization have an

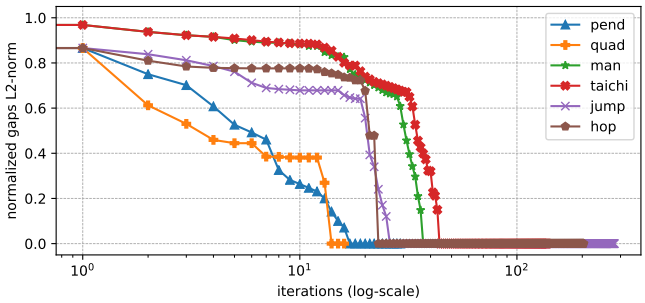

Fig. 4. Gap contraction of Box-FDDP for different optimal control problems. For all the cases, the gaps are open for the first several iterations. The gap contraction rate varies according to the accepted step-length. Smaller contraction rates, during the first iterations, appear in very nonlinear problems (taichi, man, hop, and jump), because of the larger error of the search direction.

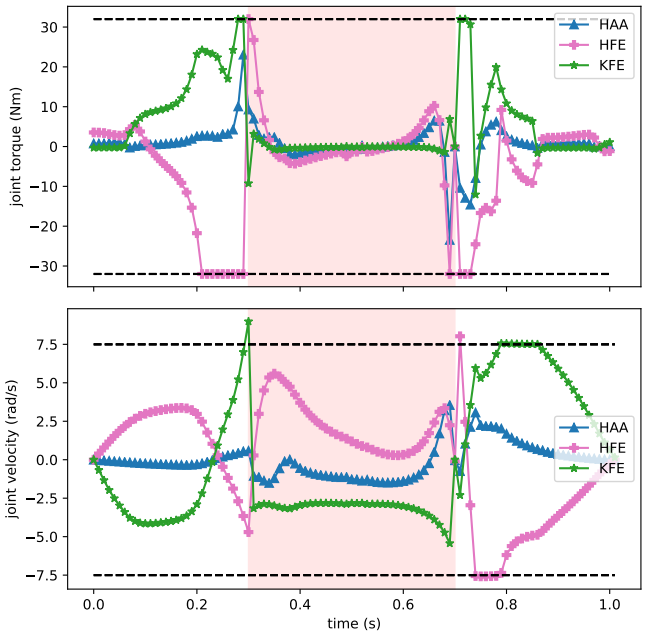

Fig. 5. Joint torques and velocity for the ANYmal jumping maneuver. (top) Generated torques of the LF joints and its limits (32 N m); (bottom) Generated velocities of the LF joint and its limits (7.5 rad/s). The red region describes the flight phase. Note that HAA, HFE, and KFE are the abduction/adduction, hip flexion/extension and knee flexion/extension joints, respectively.

important influence on the algorithm progress, mainly because DDP-based methods maintain a local quadratic approximation of the Value function. In other words, the prediction of the expected improvement is more accurate for systems with less nonlinearities and, as a result, the algorithm tends to accept bigger steps that result in higher gap reductions.

While the gaps are open, our algorithm is in feasibility-driven mode. During this phase, the cost reduction is smaller than in the control-bounded mode, in particular for very nonlinear systems (see Fig. 3 and 4). However, once the gaps are closed, a higher cost reductions often appear in very nonlinear systems.

3) Highly-dynamic maneuvers: Our algorithm can solve a wide range of motions: from unstable and consecutive hops

to aggressive and constrained jumps. In Fig. 5, we show the joint torques and velocities of a single leg for the ANYmal's jumping task (depicted in Fig. 2-b). The motion consists of three phases: jumping (0 - 300 ms), flying (300 - 700 ms), and landing (700 - 1000 ms). We reduced the real joint limits of the ANYmal robot: from 40 to 32 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. The constraint violations on the state limits appear due to the fact that we use quadratic penalization to enforce them. Nonetheless, we only encountered these violations in very constrained problems.

For the walking, trotting, pacing, and bounding gaits (reporting in the accompanying video), the Box-FDDP converges approximately with the same number of iterations achieved by the DDP solver (i.e. unconstrained case).

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

We compare the performance of Box-FDDP, Box-DDP and DDP with a squashing function for three scenarios with the IRIS quadcopter: reaching goal (goal), looping maneuver (loop), and traversing a narrow passage (narrow). We use a sigmoidal element-wise squashing function of the form:

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

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

Fig. 6 shows that Box-FDDP converges faster than the other approaches. Furthermore, the motions computed by Box-FDDP are more intuitive and with the lowest cost as reported in Table I and in accompanying video. We also observe that the squashing approach converges sooner compared to Box-DDP for the looping task. The main reason is due to the early saturation of the controls performed by Box-DDP.

In Fig. 7, we show the cost evolution for 10 different initial conditions of the reaching goal task[10]. 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 produces slightly better solutions than Box-DDP. This is in contrast to the reported results in [27], where the performance was analyzed only for LQ optimal control problem.

V. Conclusion

In this paper, we proposed a direct-indirect hybridization of the control-limited DDP algorithm (Box-FDDP). Our method discovers good solutions despite poor initial guesses thanks to infeasible iterations, which resembles a direct multiple-shooting approach. A vast range of optimal control problems demonstrate the benefits of the proposed method. Future work will focus on general inequalities constraints and model predictive control. Our implementation of Box-FDDP including all examples will be available soon.

[10]Target and initial configurations are (3, 0, 1) and (-0.3 ± 0.6, 0, 0) m, respectively.

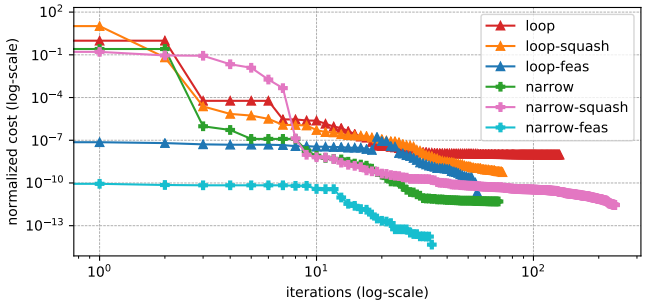

Fig. 6. Cost and convergence comparison for different quadcopter maneuvers: looping (loop) and narrow passage traversing (narrow). Box-FDDP (*-feas) outperforms both Box-DDP and DDP with squashing function (*-squash).

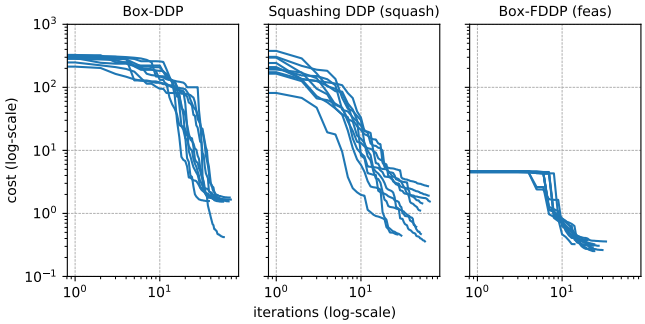

Fig. 7. Costs associated for 10 different initial conditions of reaching goal task. Box-FDDP converges earlier and with lower total cost than Box-DDP and DDP with squashing function. The performance of the squashing function approach exhibits a high dependency on the initial condition.

References

[1] 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 Robotics and Automation Letters (RA-L)*, 2017.

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

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

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

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

[6] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard. Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2018.

[7] 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.

[8] J. Carpentier and N. Mansard. Analytical Derivatives of Rigid Body Dynamics Algorithms. In *Robotics: Science and Systems (RSS)*, 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 International Conference on Robotics and Automation (ICRA)*, 2016.

[10] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[11] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber. Fast Direct Multiple Shooting Algorithms for Optimal Robot Control. In *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*. Springer Berlin Heidelberg, 2006.

[12] 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-RAS International Conference on Humanoid Robotics (Humanoids)*, 2017.

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

[14] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2018.

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

[16] A. Hereid and A. D. Ames. FROST*: Fast robot optimization and simulation toolkit. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[17] T. A. Howell, B.E. Jackson, and Z. Manchester. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[18] 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 International Conference on Intelligent Robots and Systems (IROS)*, 2015.

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

[20] 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 International Conference on Robotics and Automation (ICRA)*, 2017.

[21] 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 International Conference on Robotics and Automation (ICRA)*, 2020.

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

[23] 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 Robotics and Automation Letters (RA-L)*, 2018.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

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

[26] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[27] Y. Tassa, N. Mansard, and E. Todorov. Control-Limited Differential Dynamic Programming. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[28] Firdaus Udwadia and Robert Kalaba. A New Perspective on Constrained Motion. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 1992.

[29] 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.

[30] P.-B. Wieber. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2006.

[31] 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 Robotics and Automation Letters (RA-L)*, 2018.

[32] Z. Xie, C. K. Liu, and K. Hauser. Differential dynamic programming with nonlinear constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.