Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach

Wilson Jallet[a,b,*], Antoine Bambade[b,c], Nicolas Mansard[a] and Justin Carpentier[b]

***Abstract*— Trajectory optimization is an efficient approach for solving optimal control problems for complex robotic systems. It relies on two key components: first the transcription into a sparse nonlinear program, and second the corresponding solver to iteratively compute its solution. On one hand, differential dynamic programming (DDP) provides an efficient approach to transcribe the optimal control problem into a finite-dimensional problem while optimally exploiting the sparsity induced by time. On the other hand, augmented Lagrangian methods make it possible to formulate efficient algorithms with advanced constraint-satisfaction strategies. In this paper, we propose to combine these two approaches into an efficient optimal control algorithm accepting both equality and inequality constraints. Based on the augmented Lagrangian literature, we first derive a generic primal-dual augmented Lagrangian strategy for nonlinear problems with equality and inequality constraints. We then apply it to the dynamic programming principle to solve the value-greedy optimization problems inherent to the backward pass of DDP, which we combine with a dedicated globalization strategy, resulting in a Newton-like algorithm for solving constrained trajectory optimization problems. Contrary to previous attempts of formulating an augmented Lagrangian version of DDP, our approach exhibits adequate convergence properties without any switch in strategies. We empirically demonstrate its interest with several case-studies from the robotics literature.**

I. Introduction

In this paper, we are interested in solving constrained continuous-time optimal control problems (OCP) of the form:

$$\min_{x,u} \int_0^T \ell(t, x(t), u(t))\, dt + \ell_T(x(T)) \tag{1a}$$
$$\text{s.t. } f(t, x(t), u(t), \dot{x}(t)) = 0,\ t \in [0, T) \tag{1b}$$
$$x(0) = \bar{x}_0 \tag{1c}$$
$$h(t, x(t), u(t)) \leqslant 0 \tag{1d}$$
$$h_T(x(T)) \leqslant 0, \tag{1e}$$

where ℓ and ℓ_T are the running and terminal costs respectively, (1b) accounts for the system dynamics written as a differential-algebraic equation (DAE) (and includes the ODE case $\dot{x} = f(t, x(t), u(t))$). We denote $\mathcal{X}$ and $\mathcal{U}$ the state and control spaces, $T > 0$ the time horizon, $\bar{x}_0 \in \mathcal{X}$ the initial condition, $h(\cdot)$ and $h_T(\cdot)$ the path and terminal constraints.

For numerical resolution, the continuous-time OCP (1) must be transcribed into a finite-dimensional optimization problem (i.e., with a finite number of variables, which the continuous-time trajectories are not) [1]. Several transcriptions are possible [2]–[4]. Differential Dynamic Programming (DDP) is a particular OC algorithm which implies a direct transcription known as single shooting [5]. Popularized in robotics in the late 2000s [6], it has the advantage over other transcriptions of providing a simple formulation, optimally exploiting the sparsity of the resulting nonlinear programs while providing feedback gains at no extra cost. The corresponding transcription, extended to any constraints, reads:

$$\min_{\mathbf{x},\mathbf{u}} \sum_{k=0}^{N-1} \ell_k(x_k, u_k) + \ell_N(x_N) \tag{2a}$$
$$\text{s.t. } f_k(x_k, u_k, x_{k+1}) = 0,\ k \in [\![0, N-1]\!] \tag{2b}$$
$$x_0 = \bar{x}_0 \tag{2c}$$
$$h_k(x_k, u_k) \leqslant 0 \tag{2d}$$
$$h_N(x_N) \leqslant 0, \tag{2e}$$

where h_k, h_N, f_k are appropriate functions discretizing the dynamics and path constraints depending on the given numerical discretization scheme employed. The ℓ_k are approximations of the cost integrals $\int_{t_k}^{t_{k+1}} \ell(t, x(t), u(t))\, dt$. We use the shorthands $\mathbf{x} \overset{\text{def}}{=} (x_0, \ldots, x_N)$ and $\mathbf{u} \overset{\text{def}}{=} (u_0, \ldots, u_{N-1})$ for the discretized state and control trajectories.

While the nominal DDP algorithm is not able to handle path constraints, implicit integrators or multiple-shooting stabilization, several improvements have been proposed over the years to equip it with these properties. In this paper, we focus on the handling of equality and inequality constraints, and we first review previous work focusing on it.

A first subcase of interest only considers OCP with control bounds, which can be handled by a projected quasi-Newton approach [7]. Several other projection-based formulations have then been proposed to extend DDP [8], [9], none of which have been shown to be robust enough to be widely adopted in robotics. To account fro inequality constraints, interior-point methods [10], [11] have also been recently investigated; however, these do not allow for easy warm-starting [3] which is unsuitable for online optimization and application to model-predictive control (MPC) [12], [13].

In the past few years, augmented Lagrangian approaches have emerged as a suitable solution for solving constrained

[a] LAAS-CNRS, 7 Avenue du Colonel Roche, F-31400 Toulouse, France
[b] Inria, Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, Paris, France
[c] ENPC, France, *corresponding author: wjallet@laas.fr

This work was supported in part by the HPC resources from GENCI-IDRIS (Grant AD011011342), the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute) and ANR-19- P3IA-000 (ANITI 3IA Institute), Louis Vuitton ENS Chair on Artificial Intelligence, and the European project MEMMO (Grant 780684).

trajectory optimization problems [14]. As argued later in this paper, it offers many of the good properties that we need for trajectory optimization: super-linear convergence or even more quadratic convergence, stability, ability to warm-start, and so on. Yet the first attempt to write dedicated OCP solvers based on augmented Lagrangians exhibited poor convergence properties. Thereby, further refinement using a projection in a two-stage approach had to be introduced in the solver ALTRO [15]. The penalty function used in ALTRO was then recognized to be irregular and discarded in [16], which introduces a switch to an SQP formulation to converge to a higher precision.

A key idea that we exploit in this paper is to introduce the augmented Lagrangian formulation directly in the backward pass, to solve the value-greedy problems while directly considering the constraints, as initially proposed for multi-phase constrained problems [17]. This enables us to obtain better numerical accuracy for equality-constrained problems, by stabilizing the backward pass using a primal-dual system of equations to compute the control and multipliers together [18], and a monotonic update of the penalty parameter derived from the bound-constrained Lagragian (BCL) [19] strategy. Their method converges reliably to good numerical accuracy. We have recently extended this formulation to also account for the dynamics and other equality constraints using a primal-dual augmented Lagrangian, allowing for the inclusion of infeasible initialization and implicit integrators [20].

In this paper, we introduce a complete augmented Lagrangian DDP algorithm for handling both equality and inequality constraints, and validate it on several real-size robotic scenarios. We first introduce in Sec. II a primal-dual algorithm, rooted in the nonlinear programming literature [21], to handle generic nonlinear optimization problems (NLPs). We then adapt it to the specific case of OCPs of the form (2) in Sec. III. It results in an overall second-order quasi-Newton-like algorithm with good convergence properties for solving constrained trajectory optimization problems. We finally benchmark our method in Sec. IV on various standard case studies from the robotics literature. A companion video is available[1].

II. The primal-dual augmented Lagrangian method for constrained optimization

This section introduces our augmented Lagrangian approach to solve constrained nonlinear optimization problems (NLP) of the form:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \; f(x) \\ &\text{s.t. } c(x)=0,\; h(x)\leqslant 0, \end{aligned} \tag{3}$$

where c and h stands for equality and inequality constraints respectively. We then adapt this approach in Sec. III to the case of trajectory optimization. While many augmented Lagrangian approaches have been introduced in the optimization literature [22], most of them rely on alternating between primal solving and dual updates. In this work, we propose instead to compute combined primal-dual steps by taking inspiration from the work of Gill and Robinson in [21], which we extend by also considering inequality constraints and by connecting it to the proximal method of multipliers (PMM) [23] that we use to for numerical robustness. We discuss these contributions in more detail at the end of this section.

A. Optimality conditions

The Lagrangian $\mathcal{L}$ associated with (3) is defined by:

$$\mathcal{L}(x,\lambda,\nu) = f(x) + \lambda^\top c(x) + \nu^\top h(x),\;\; \lambda\in\mathbb{R}^{n_e}, \nu\in\mathbb{R}^{n_i}_+ \tag{4}$$

A saddle point of $\mathcal{L}$ is a solution of (3). This leads to the Karush-Kuhn-Tucker (KKT) necessary conditions [24] for ensuring a primal-dual point (x,λ,ν) to be optimal:

$$\begin{aligned} \nabla_x\mathcal{L}(x,\lambda,\nu) &= 0, \\ c(x)=0 \text{ and } h(x) &\leqslant 0, \\ \nu\geqslant 0 \text{ and } h(x)^\top\nu &= 0. \end{aligned} \tag{KKT}$$

In practice, we search for a triplet (x,λ,ν) satisfying these optimality conditions (KKT) up to a certain level of predefined accuracy $\epsilon_{\text{abs}} > 0$, leading us to the following natural absolute *stopping criterion*:

$$\begin{cases} \|\nabla_x\mathcal{L}(x,\lambda,\nu)\|_\infty \leqslant \epsilon_{\text{abs}}, \\ \|(c(x),[h(x)]_+)\|_\infty \leqslant \epsilon_{\text{abs}}, \end{cases} \tag{5}$$

where $[z]_+$ denotes the projection of $z\in\mathbb{R}^{n_z}$ on $\mathbb{R}^{n_z}_+$.

B. Equality constrained nonlinear programming problems

In this section, we provide a high-level overview on the primal-dual augmented Lagrangian (PDAL) method. It is closely related to the probably even more famous Method of Multipliers (MM) [25, Chapter 2], which we review first in the context of purely equality-constrained NLPs.

Primal-dual augmented Lagrangian. The PDAL function $\mathcal{L}^A_\mu$ [21, Section 3] is defined by augmenting the standard Lagrangian $\mathcal{L}$ (4) with two squared ℓ_2 penalties:

$$\begin{aligned} \mathcal{M}^A_\mu(x,\lambda;\lambda_e) \overset{\text{def}}{=} \mathcal{L}(x,\lambda_e,0) &+ \tfrac{1}{2\mu_{\text{e}}}\|c(x)\|_2^2 \\ &+ \tfrac{1}{2\mu_{\text{e}}}\|c(x)+\mu_e(\lambda_e-\lambda)\|_2^2, \end{aligned} \tag{6}$$

where $\mu_{\text{e}} > 0$ is a scalar[2]. The PDAL method then searches a sequence of iterates approximately minimizing (6) [28]:

$$x_{l+1},\lambda_{l+1} \approx_{\omega_l} \min_{x,\lambda} \mathcal{M}^A_\mu(x,\lambda;\lambda_l), \tag{7}$$

where $\approx_{\omega_l}$ stands for requiring (x_{l+1},λ_{l+1}) to be an ω_l-approximate solution to subproblem (7). The approximation is controlled via the following condition:

$$\|r_l(x_{l+1},\lambda_{l+1})\|_\infty \leqslant \omega_l, \tag{8}$$

where r_l accounts for the optimality conditions at iteration l: if $\|r_l(x,\lambda)\|_\infty \leqslant \epsilon_{\text{abs}}$ then (x,λ) is a solution to (7) at precision ϵ_{abs}. The formula for r_l will be specified in the

[1] https://peertube.laas.fr/videos/watch/dfeca51c-c2cf-468a-b46a-86f808e9a561

[2] Some authors associate a penalty parameter to each constraint. In this case, the penalty parameters μ_{e} is a matrix Σ_λ [26], [27].

sequel. The first subproblems are solved coarsely, since a precise solution is not required when the multiplier estimates λ^0 are far from the optimal dual solution: this avoids unnecessary computation.

To enforce the generated sequence (x_l, λ_l) to converge to a local solution of NLP problem (3), we must address two important aspects: (i) computing suitable iterates (x_{l+1}, λ_{l+1}) satisfying (8) efficiently; (ii) choosing appropriate rules for scheduling ω_l (ω_l should decrease) and adequately increasing μ_{e} (as shown by the theory, μ_{e} should be increased over the iterations, but a too large value may drastically impact the overall numerical stability [22]).

In the last two paragraphs of section II-B, the problem (i) of finding suitable approximations for the subproblems is handled in the next paragraph. Then we review in the last paragraph the BCL globalization strategy (originating from [29]) for dealing with (ii).

Primal-dual Newton descent. The stationarity condition of (7) is :

$$\nabla \mathcal{M}_{\mu}^{A}(x, \lambda; \lambda_l) = 0. \tag{9}$$

Iterates (x_{l+1}, λ_{l+1}) satisfying (9) at precision ω_l can be derived using a quasi-Newton descent [21], which iteration at $t+1$ starting from $(\hat{x}_l^0, \hat{\lambda}_l^0) = (x_l, \lambda_l)$ reads:

$$\begin{bmatrix} H_l + \frac{1}{\mu_e} J_c^\top J_c & -J_c^\top \\ -J_c & \mu_e I \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}_{\mu}^{A}(\hat{x}_l^t, \hat{\lambda}_l^t; \lambda_l) \\ \nabla_\lambda \mathcal{L}_{\mu}^{A}(\hat{x}_l^t, \hat{\lambda}_l^t; \lambda_l) \end{bmatrix}, \tag{10}$$

with:

$$\hat{x}_l^{t+1} = \hat{x}_l^t + \delta x, \quad \hat{\lambda}_l^{t+1} = \hat{\lambda}_l^t + \delta \lambda, \tag{11}$$

and where H_l is the Lagrangian Hessian $\nabla_x^2 \mathcal{L}(\hat{x}_l^t, 2\pi_l(\hat{x}_l^t) - \hat{\lambda}_l^t, 0)$ or an approximation thereof, with $\pi_l(x) \overset{\text{def}}{=} \lambda_l + \frac{1}{\mu_{\mathrm{e}}} c(x)$ (following the notation in [21]), and J_c the constraint Jacobian matrix at $\hat{x}_l^t$.

There are two conflicting goals to balance in this iterative process. First, the smaller the value of μ_{e} the faster the convergence, as $\frac{1}{\mu_e}$ penalizes the constraints. Second, as μ_{e} gets smaller, the conditioning of $(H_l + \frac{1}{\mu_{\mathrm{e}}} J_c^T J_c)$ gets worse, thereby harming the numerical robustness of the approach, in particular when J_c has a large condition number.

Fortunately, the linear system (10) can be rewritten in the following equivalent form:

$$\begin{bmatrix} H_l & J_c^\top \\ J_c & -\mu_{\mathrm{e}} I \end{bmatrix} \begin{bmatrix} \delta x \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x \mathcal{L}(\hat{x}_l^t, \hat{\lambda}_l^t, 0) \\ c(\hat{x}_l^t) + \mu_{\mathrm{e}}(\lambda_l - \hat{\lambda}_l^t), \end{bmatrix}, \tag{12}$$

which shows that the PDAL method is closely related to the method of multipliers (MM) [25, Chapter 2]. Indeed, (12) implies that the sequence of iterates (x_l, λ_l) approximate those of a proximal-point method applied to the dual of (4):

$$x_{l+1}, \lambda_{l+1} \approx_{\omega_l} \min_x \max_\lambda \mathcal{L}(x, \lambda, 0) - \tfrac{\mu_{\mathrm{e}}}{2} \|\lambda - \lambda_l\|_2^2. \tag{13}$$

Hence, μ_{e} is the inverse of the step-size of the equivalent MM, and it directly calibrates the convergence speed of the approach (see [23, Section 4] for details). Moreover, this linear system involves a matrix that is always nonsingular thanks to the regularization terms $-\frac{\mu_{\mathrm{e}}}{2} \|\lambda - \lambda_l\|_2^2$. In other words, the problem (12) is always well-defined in the iterative process. Such linear systems are also better conditioned than (10) [22, Section 17.1].

Finally, (12) implies that the sequence $(\hat{x}_l^t, \hat{\lambda}_l^t)_{t \geqslant 0}$ converges to a pair (x^*, λ^*) satisfying the following optimality conditions:

$$\begin{bmatrix} \nabla_x \mathcal{L}(x^*, \lambda^*, 0) \\ c(x^*) + \mu_{\mathrm{e}}(\lambda_l - \lambda^*) \end{bmatrix} = 0. \tag{14}$$

Hence, we choose the optimality criterion function r_l to be:

$$r_l(x, \lambda) \overset{\text{def}}{=} \begin{bmatrix} \nabla_x \mathcal{L}(x, \lambda, 0) \\ c(x) + \mu_{\mathrm{e}}(\lambda_l - \lambda) \end{bmatrix}. \tag{15}$$

The globalization strategy. For fixing the hyper-parameters (tolerance on subproblems ω_l, step-sizes μ_{e}), we rely on BCL (see [19] and [22, Algorithm 17.4]) which has been proved to perform well in advanced optimization packages such as LANCELOT [30] and also in robotics for solving constrained optimal control problems [16], [31].

The main idea underlying BCL consists in updating the dual variables λ_l from (13) only when the corresponding primal feasibility (denoted by η_l hereafter) is small enough. More precisely, we use a second sequence of tolerances denoted by ϵ_l (which we also tune within the BCL strategy) and update the dual variables only when $\eta_{l+1} \leqslant \epsilon_l$, where η_{l+1} denotes the primal infeasibility as follows:

$$\eta_{l+1} \overset{\text{def}}{=} \|c(x_{l+1})\|_\infty. \tag{16}$$

It remains to explain how the BCL strategy chooses appropriate values for the hyper-parameters ω_l, ϵ_l and μ_e. As for the update of the dual variables, it proceeds in two stages:

- **If** $\eta_{l+1} < \epsilon_l$**:** the primal feasibility is good enough, we thus keep the constraint penalization parameters as is.
- **Otherwise:** the primal infeasibility is too large, we thus increase quadratic penalization terms on the constraints for the subsequent subproblem (7).

Concerning the accuracy parameters ω_l and ϵ_l, the update rules are more technical and the motivation underlying those choices is to ensure global convergence: an exponential-decay type update when primal feasibility is good enough, and see [29, Lemma 4.1] for when the infeasibility is too large. The detailed strategy is summarized in Algorithm 1 for the general case (including inequalities).

C. Extension to inequality constrained nonlinear programs

As we will see, our approach developed for tackling equality constraints easily extends to the general case. Indeed, as we will see the PDAL function only changes in a subtle way for taking into account inequality constraints. As a result, it also impacts how the minimization procedure must be realized.

Generalized primal-dual merit function. In the general setup, the PDAL function can be framed in its equality constrained form introducing a slack variable $z \leqslant 0$ satisfying the new equality constraint:

$$h(x) - z = 0. \tag{17}$$

Hence, the generalized PDAL function reads:

$$\begin{aligned}\mathcal{L}_\mu^A(x,\lambda,\nu,z;\lambda_l,\nu_l) &\stackrel{\text{def}}{=} \mathcal{L}(x,\lambda_l,\nu_l) + \tfrac{1}{2\mu_\text{e}}\|c(x)\|_2^2 \\ &+ \tfrac{1}{2\mu_\text{e}}\|c(x)+\mu_e(\lambda_l-\lambda)\|_2^2 + \tfrac{1}{2\mu_\text{i}}\|h(x)-z\|_2^2 \\ &+ \tfrac{1}{2\mu_\text{i}}\|h(x)-z+\mu_i\nu_l-\mu_i\nu\|_2^2 + g(z).\end{aligned} \tag{18}$$

g is the (component-wise) indicator function related to $z \leqslant 0$:

$$g(z) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } z_i \leqslant 0, i \in [1,n_i], \\ +\infty & \text{otherwise.}\end{cases}$$

The minimization of (18) w.r.t. x, λ, ν or z variables commutes. Considering the problem structure and following ideas from [32], it can be shown that z and ν can be directly deduced as functions of x:

$$\begin{aligned}\hat{z}(x,\nu_l),\hat{\nu}(x,\nu_l) &\stackrel{\text{def}}{=} \arg\min_{z,\nu} \mathcal{L}_\mu^A(x,\lambda,\nu,z;\lambda_l,\nu_l), \\ \hat{z}(x,\nu_l) &= [h(x)+\mu_\text{i}\nu_l]_-, \\ \hat{\nu}(x,\nu_l) &= \left[\tfrac{1}{\mu_\text{i}}h(x)+\nu_l\right]_+.\end{aligned} \tag{19}$$

The minimization problem can thus be reduced to:

$$\begin{aligned}&\min_{x,\lambda,\nu,z} \mathcal{L}_\mu^A(x,\lambda,\nu,z;\lambda_l,\nu_l) \\ &= \min_{x,\lambda} \mathcal{L}_\mu^A(x,\lambda,\hat{\nu}(x,\nu_l),\hat{z}(x,\nu_l);\lambda_l,\nu_l).\end{aligned} \tag{20}$$

Yet, we choose to maintain the ν variable relaxed in the minimization procedure (20) as it enables us to preserve similar well conditioned linear systems and stopping criterion derived in (12) and (14). Consequently, the generalized merit function corresponds to:

$$\begin{aligned}&\mathcal{M}_\mu(x,\lambda,\nu;\lambda_l,\nu_l) \stackrel{\text{def}}{=} \mathcal{L}_\mu^A(x,\lambda,\nu,\hat{z}(x,\nu_l);\lambda_l,\nu_l) \\ &= \mathcal{L}(x,\lambda_l,0) + \tfrac{1}{2\mu_\text{e}}\|c(x)\|_2^2 + \tfrac{1}{2\mu_\text{e}}\left\|c(x)+\mu_\text{e}(\lambda_l-\lambda)\right\|_2^2 \\ &+ \tfrac{1}{2\mu_\text{i}}\left\|\,[h(x)+\mu_\text{i}\nu_l]_+\,\right\|_2^2 + \tfrac{1}{2\mu_\text{i}}\left\|\,[h(x)+\mu_\text{i}\nu_l]_+ - \mu_\text{i}\nu\right\|_2^2.\end{aligned} \tag{21}$$

For ensuring better regularization w.r.t. the primal variable x, following the PMM [23], we finally consider the following generalized primal-dual merit function:

$$\mathcal{M}_{\mu,\rho}(x,\lambda,\nu;x_l,\lambda_l,\nu_l) \stackrel{\text{def}}{=} \mathcal{M}_\mu(x,\lambda,\nu;\lambda_l,\nu_l) + \frac{\rho}{2}\|x-x_l\|_2^2, \tag{22}$$

with $\rho > 0$ a proximal parameter.

Semi-smooth Newton step with line-search procedure. Contrary to the equality-constrained case, (22) now corresponds to a semi-smooth function (due to the presence of the positive orthant projection operators $[\cdot]_+$). It should thus be minimized using a semi-smooth quasi-Newton iterative procedure [33, Chapter 1], [22, Chapter 6]. To ensure convergence, the procedure must have a line-search scheme[3] over the semi-smooth convex primal-dual merit function (22) to find an adequate step size along the primal-dual Newton direction, following an approach similar to (12), while also considering the change of active inequality-constraints defined by $\mathcal{A}_l(x)$:

$$\mathcal{A}_l(x) \stackrel{\text{def}}{=} \{j \mid (\nu_l + \mu_i h_j(x)) \geqslant 0\}, \tag{23}$$

where $\mathcal{A}_l(x)$ is the shifted active-set of the l-th subproblem at point x. This definition of shifted active set differs from existing augmented Lagrangian-based optimal control methods in robotics which defines the active-set by the condition $h_j(x) \geqslant 0$, as done in [15], [34].

[3]Different line-search schemes can be used to minimize the merit function (22) through a semi-smooth quasi-Newton iterative procedure.

Algorithm 1: PDAL Method for constrained optimization

Inputs:
- initial states: x_0, λ_0, ν_0,
- initial parameters: ϵ_0, ω_0, ρ, μ_e, $\mu_i > 0$,
- hyper-parameters: $\mu_f < 1, \alpha_\text{bcl} \in (0,1), \beta_\text{bcl} \in (0,1)$, $\underline{\mu_\text{i}}, \underline{\mu_\text{e}} > 0$.

while *Stopping criterion* (5) *not satisfied* **do**
- Compute $(\widetilde{x}_{l+1}, \widetilde{\lambda}_{l+1}, \widetilde{\nu}_{l+1})$ satisfying (26) using II-C ;
- $x_{l+1} = \widetilde{x}_{l+1}$;
- **if** $\eta_{l+1} < \epsilon_l$ **then**
  - $\epsilon_{l+1} = \epsilon_l \mu_\text{i}^{\beta_\text{bcl}}$; $\omega_{l+1} = \omega_l \mu_\text{i}$;
  - $\lambda_{l+1} = 2\hat{\lambda}(x_{l+1},\lambda_l) - \widetilde{\lambda}_{l+1}$;
  - $\nu_{l+1} = [2\hat{\nu}(x_{l+1},\nu_l) - \widetilde{\nu}_{l+1}]_+$;
- **else**
  - $\mu_\text{i} \leftarrow \max(\underline{\mu_\text{i}}, \mu_f\mu_\text{i}), \mu_e \leftarrow \max(\underline{\mu_\text{e}}, \mu_f\mu_\text{e})$;
  - $\epsilon_{l+1} = \epsilon_0 \mu_\text{i}^{\alpha_\text{bcl}}$; $\omega_{l+1} = \omega_0\mu_\text{i}$;
  - $\lambda_{l+1} = \lambda_l$; $\nu_{l+1} = \nu_l$;
- **end**
- $l \leftarrow l+1$;

end

Output: A (x_l,λ_l,ν_l) satisfying the ϵ_abs-approximation criterion (5) for problem (3).

D. Final algorithm

Once a local ω_l primal-dual solution (x^*,λ^*,ν^*) minimizing (22) is found, for better numerical precision, we follow the Lagrange multiplier update rule introduced in [28, Section 4]:

$$\begin{aligned}\lambda_{l+1} &= 2\hat{\lambda}(x^*,\lambda_l) - \lambda^*, \\ \nu_{l+1} &= [2\hat{\nu}(x^*,\nu_l) - \nu^*]_+,\end{aligned} \tag{24}$$

where $\hat{\nu}(x,\nu_l)$ is defined from the derivation of our generalized merit function in (19), and $\hat{\lambda}(x,\lambda_l)$ comes from the classic multiplier update rule (similar to $\hat{\nu}(x,\nu_l)$ without projections) [28]. Hence, the measure of the convergence towards the optimality conditions captured by r_l (15) can be more generally defined as follows:

$$r_l(x,\lambda,\nu) \stackrel{\text{def}}{=} \begin{bmatrix} \nabla_x\mathcal{L}(x,\lambda,\nu) + \rho(x-x_l) \\ \mu_\text{e}(\hat{\lambda}(x,\lambda_l)-\lambda) \\ \mu_\text{i}(\hat{\nu}(x,\nu_l)-\nu)\end{bmatrix}. \tag{25}$$

The generalized inner-loop exit condition thus reads:

$$\|r_l(x,\lambda,\nu)\|_\infty \leqslant \omega_l. \tag{26}$$

The primal feasibility also generalizes as:

$$\eta_{l+1} \stackrel{\text{def}}{=} \|(c(x_{l+1}), [h(x_{l+1})]_+)\|_\infty. \tag{27}$$

Algorithm 1 summarizes our approach for solving NLPs.

E. Key novelties of Algorithm 1

Algorithm 1 differs from [21] for two main aspects. First, we show in the equality-constrained case that the linear systems involved in the (quasi-)Newton steps are equivalent to a better-conditioned linear system originating from the proximal method of multipliers [23]. For this reason, we use this equivalent saddle-point system formulation and its associated stopping criterion to enforce the overall numerical stability of the approach. Second, we extend the PDAL function from [21] to account for inequality constraints by introducing a new merit function that does not require any slack variables.

The resulting algorithm is a generic NLP solver which is a contribution in itself, with direct application in optimization for robotics e.g. [35]. As our aim is to design a constrained OCP solver, we have left evaluation of this generic solver's performance for future work, and we directly jump to its adaptation to dynamic programming.

III. Primal-Dual Augmented Lagrangian for Constrained Differential Dynamic Programming

In this section, we extend the differential dynamic programming framework accounting for equality constraints [18] and implicit dynamics [20] to the case of inequality constraints using the PDAL introduced in Sec. II.

A. Relaxation of the Bellman equation.

In [20], we show that applying the MM to (2) leads to a relaxation of the Bellman equation, in the equality-constrained case. We now extend this idea to the inequality-constrained case. Indeed, the discrete-time problem (2) also satisfies a dynamic programming equation in the inequality-constrained case. The value function for the subproblem at time k satisfies the Bellman relation:

$$\begin{aligned} V_k(x) = \min_{u,x'} \ell_k(x,u) + V_{k+1}(x') \\ \text{s.t. } f_k(x,u,x') = 0 \text{ and } h_k(x,u) \leqslant 0. \end{aligned} \tag{28}$$

The optimality conditions for this Bellman equation involve a Lagrangian function of the form:

$$\begin{aligned} \mathcal{L}_k(x,u,x',\lambda,\nu) = \ell_k(x,u) + V_{k+1}(x') \\ + \lambda^\top f_k(x,u,x') + \nu^\top h_k(x,u). \end{aligned} \tag{29}$$

The last term in this Lagrangian, relating to the inequality constraint, will appear in the KKT conditions of the Bellman equation and influence the sensitivities with respect to x. Following the PDAL method from Sec. II, we can define an augmented *primal-dual Q-function* modelled after (21) which reads, considering multiplier estimates (λ_l, ν_l), $l \in \mathbb{N}$:

$$\begin{aligned} &Q^l_{\mu,k}(x,u,x',\lambda,\nu) = \ell_k(x,u) + V_{k+1}(x') \\ &+ \tfrac{\mu_\text{e}}{2}(\|\tfrac{1}{\mu_\text{e}} f_k(x,u,x') + \lambda_l\|^2 + \|\tfrac{1}{\mu_\text{e}} f_k(x,u,x') + \lambda_l - \lambda\|^2) \\ &+ \tfrac{\mu_\text{i}}{2}(\|[\nu_l + \tfrac{1}{\mu_\text{i}} h_k(x,u)]_+\|^2 + \|[\nu_l + \tfrac{1}{\mu_\text{i}} h_k(x,u)]_+ - \nu\|^2). \end{aligned} \tag{30}$$

Then, the minimization in the Bellman equation is relaxed to the following augmented Lagrangian iteration:

$$V^l_k(x) = \min_{u,x'\lambda,\nu} Q^l_{\mu,k}(x,u,x',\lambda,\nu), \tag{31}$$

with the boundary condition $V^l(x) = \ell_\text{f}(x)$. This dynamic programming equation can be seen as a relaxation of the classical Bellman equation (28) using the primal-dual merit function – including the dynamics $f_k(x,u,x')$ in this relaxation leads to a multiple-shooting formulation. Indeed, assuming the multipliers estimates (λ_l, ν_l) are optimal multipliers associated with (28), then any minimizer $(\bar{u}, \bar{x}', \bar{\lambda}, \bar{\nu})$ of (31) also satisfies the optimality conditions for (28).

B. Backward and forward passes

As outlined in the previous section, the semi-smooth (quasi-)Newton descent direction for Q^l can be recovered from the system of equations:

$$\mathcal{K}_\mu \begin{bmatrix} \delta u \\ \delta x' \\ \delta\lambda \\ [\delta\nu]_\mathcal{A} \end{bmatrix} = - \begin{bmatrix} Q_u + Q_{ux}\delta x \\ Q_{x'} + Q_{x'x}\delta x \\ f + f_x \delta x + \mu_\text{e}(\lambda_l - \lambda) \\ [h + h_x \delta x + \mu_\text{i}(\nu_l - \nu)]_\mathcal{A} \end{bmatrix}, \tag{32}$$

where:

$$\mathcal{K}_\mu \stackrel{\text{def}}{=} \begin{bmatrix} Q_{uu} & Q_{ux'} & f_u^\top & [h_u]_\mathcal{A}^\top \\ Q_{x'u} & Q_{x'x'} & f_{x'}^\top & [h_{x'}]_\mathcal{A}^\top \\ f_u & f_{x'} & -\mu_\text{e} I & \\ [h_u]_\mathcal{A} & [h_{x'}]_\mathcal{A} & & -\mu_\text{i} I \end{bmatrix}, \tag{33}$$

is a regularized KKT matrix. It is similar to the matrix derived in [20], with an additional block covering the active set of inequality constraints, denoted by $[\cdot]_\mathcal{A}$. Subscripted symbols (e.g. $f_x, h_u \dots$) denote partial derivatives. We switch convention from [20], where μ is the reciprocal of the parameters $(\mu_\text{e}, \mu_\text{i})$ we use here.

Since the previous state deviation δx is unknown but the *r.h.s.* of (32) is linear in that parameter, we can recover the solution from the sensitivities, which satisfy:

$$\mathcal{K}_\mu \begin{bmatrix} k & K \\ a & A \\ \xi & \Xi \\ \zeta_\mathcal{A} & Z_\mathcal{A} \end{bmatrix} = - \begin{bmatrix} Q_u & Q_{ux} \\ Q_{x'} & Q_{x'x} \\ f + \mu_\text{e}(\lambda_e - \lambda) & f_x \\ [h + \mu_\text{i}(\nu_l - \nu)]_\mathcal{A} & [h_x]_\mathcal{A} \end{bmatrix}. \tag{34}$$

The step is recovered as:

$$\begin{aligned} &\delta u = k + K\delta x, \quad \delta x' = a + A\delta x \\ &\delta\lambda = \xi + \Xi\delta x, \quad \delta\nu = -[\nu]_{\mathcal{A}^c} + \zeta_\mathcal{A} + Z_\mathcal{A}\delta x. \end{aligned} \tag{35}$$

In practice, the system (34) is solved by an $LDL^\top$ Cholesky factorization of the KKT matrix $\mathcal{K}_\mu$.

Forward pass and linear rollout. Similarly to [20], the primal-dual step is recovered by a linear rollout over (35):

$$\begin{aligned} &\delta u_t = k_t + K_t \delta x_t, \ \delta x_{t+1} = a_t + A_t \delta x_t \\ &\delta\lambda_{t+1} = \xi_t + \Xi_t \delta x_t, \ \delta\nu_{t+1} = -[\nu_{t+1}]_{\mathcal{A}^c} + \zeta_{\mathcal{A},t} + Z_{\mathcal{A},t}\delta x_t. \end{aligned} \tag{36}$$

The initial step over $(\delta x_0, \delta\lambda_0, \delta\nu_0)$ is associated with the value function, equality and inequality constraints at $k = 0$.

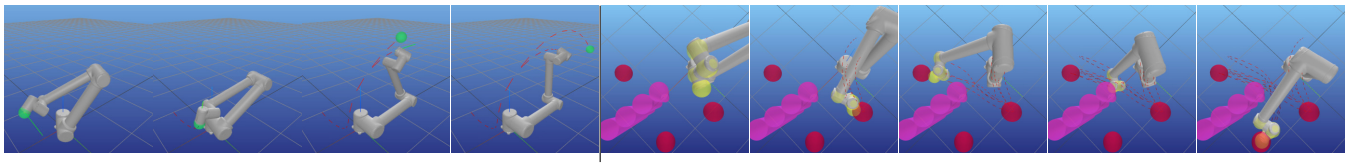

Fig. 1. **Left.** Throwing motion on UR10. The ball is the yellow, then green sphere. **Right.** UR10 reach task. The yellow spheres around the end-effector and wrist links do not collide with the purple cylinders, and the waypoints are reached at the specified times.

C. Convergence and globalization strategy

As discussed in Sec. II and following the approach proposed in [18], [20], we use a BCL strategy as an outer loop to automatically update the parameters μ and ρ and the multipliers estimates (λ_l, ν_l) according to the progress made on the primal and dual feasibility. We refer to [20] to see how BCL is used within the constrained DDP. We also use a backtracking line-search procedure to compute a step length $\alpha > 0$ at each iteration of the algorithm after the linear rollout. This line-search relies on the assumption that the direction $\delta\mathbf{w} = (\delta\mathbf{x}, \delta\mathbf{u}, \delta\boldsymbol{\lambda}, \delta\boldsymbol{\nu})$ is a descent direction satisfying $\nabla\mathcal{M}_{\mu,\rho}^{\top}\delta\mathbf{w} < 0$. Denoting $\phi(\alpha) = \mathcal{M}_{\mu,\rho}(\mathbf{w} + \alpha\delta\mathbf{w}; \mathbf{w}_l)$, our Armijo backtracking procedure looks for the first $k \in \mathbb{N}$ such that $\phi(t^k) \leqslant \phi(0) + c_1 t^k \phi'(0)$ and sets $\alpha = t^k$. To ensure the descent condition, we play on the proximal parameters $\rho_l > 0$ in the outer BCL loop, and have a heuristic similar to [36] and [6] to control the inertias of the regularized KKT matrices (33), which is central to obtain good convergence behavior. Our stopping criteria is the same as in the constrained optimization framework outlined in Alg. 1.

IV. Experiments

For experimental validation our approach, we extend the numerical optimal control framework of [20], written in Python, which relies on the Pinocchio rigid-body dynamics library [37] for providing analytical derivatives [38] and NumPy [39] for linear algebra. Because it is in Python, we do not provide CPU timings against existing implementations.

A. Bound-constrained problems

Bound-constrained LQR. The first system we test is the simple linear-quadratic regulator (LQR) with bound constraints, of the form:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} \frac{1}{2} x_k^{\top} Q x_k + \frac{1}{2} u_k^{\top} R u_k + \frac{1}{2} x_N^{\top} Q_N x_N \\ \text{s.t. } & x_{k+1} = Ax_k + Bu_k + c, \;\; k \leqslant N-1 \\ & x_0 = \bar{x}_0, \;\; -\bar{u} \leqslant u_k \leqslant \bar{u} \end{aligned} \tag{37}$$

where Q, Q_N and R are positive semi-definite matrices, $\bar{u} \in \overline{\mathbb{R}}_{+}^{n_u}$ are the control bounds. This problem is a convex quadratic program (QP), which can be solved with classical QP solvers. We test a few configurations for the problem parameters $(A, c, \bar{u})$, leading to the results in Fig. 2 and 3. For bound-constrained LQR, the proposed method takes about ten iterations to converge to an optimal solution with precision $\epsilon = 10^{-8}$.

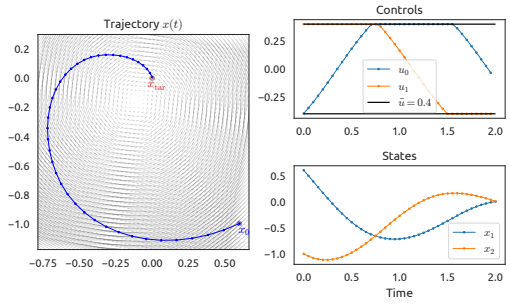

Fig. 2. Rotational system: A approximates a continuous-time system with matrix $A_c = \left[\begin{smallmatrix} 0 & 2 \\ -2 & 0 \end{smallmatrix}\right]$, $c = (0.3, -0.2)$, control bound $\bar{u} = 0.4$. In spite of the control bounds, which saturate, the target is reached.

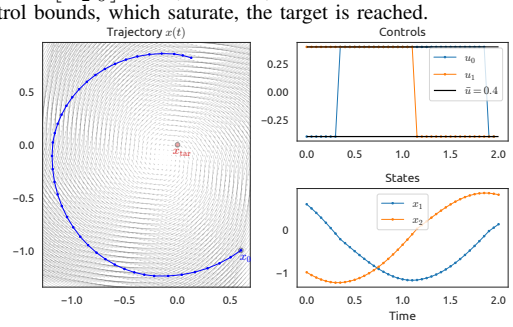

Fig. 3. The continuous time system is $A_c = \left[\begin{smallmatrix} 0.4 & 2 \\ -2 & 0.4 \end{smallmatrix}\right]$, c is the same, the target is located near a repulsive equilibrium. Even with the same bound $\bar{u} = 0.4$, the target is not reached. The obtained control is a bang-bang control.

Car parking, as proposed in [7, IV.B.]. The car dynamics is defined by its state variables (x, y, θ, v), the goal is to steer the car to the state $(0, 0, 0, 0)$. The control inputs are the front wheel acceleration $a \in \mathbb{R}$ and angle ω, with bounds $|a| \leqslant 10\,\mathrm{m/s^2}, |\omega| \leqslant 0.5\,\mathrm{s^{-1}}$. Figure 4 illustrates the resulting trajectory with comments. Following [7], the system makes the distinction between the initial angles $\frac{3\pi}{2}$ and $-\frac{\pi}{2}$, which (interestingly for the interest of the benchmark) forces the solver to find more commutations. We used initial penalty parameters $\mu_0 = 100, \rho_0 = 10^{-5}$, and convergence threshold $\epsilon = 2.10^{-4}$ (no convergence threshold was given in [7]). The timestep is $dt = 0.03\,\mathrm{s}$ and horizon $T = 15\,\mathrm{s}$. The problem converges to an optimal solution in 62 iterations.

UR5 – throwing task. The goal of this task is to throw a ball at a target velocity $\bar{v}$ at a time half-way through the horizon $T = 1\,\mathrm{s}$ (with a minimum velocity in the z direction). We constrain the elbow frame to be above ground, the end-effector stay within a box, along with joint velocity and torque limits. The dynamics are integrated using a second order Runge-Kutta scheme with timestep $dt = 0.05\,\mathrm{s}$. The state and control trajectories satisfy the bound constraints as depicted in in Fig. 5. The robot motion is illustrated in Fig. 1.

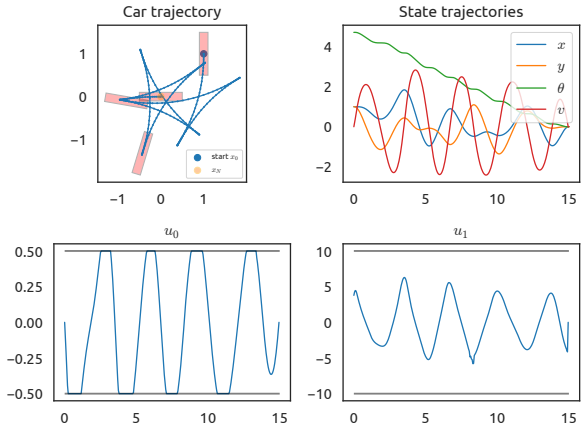

Fig. 4. Solution of the car parking task. The starting state is $(1, 1, \frac{3\pi}{2}, 0)$. The turn control $u_0 = \omega$ often saturates, causing the policy to go backwards to turn further; due to the parametrization with angle θ, a lot more turning is required.

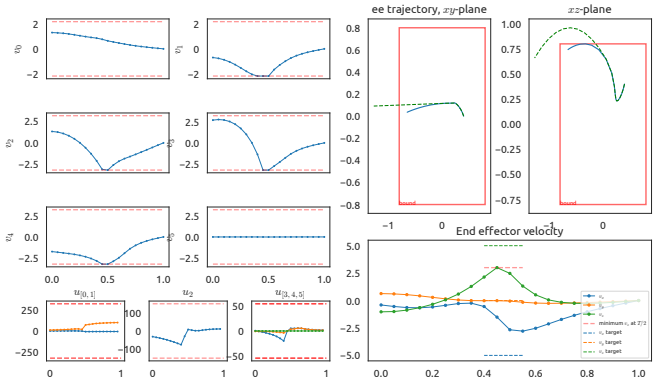

Fig. 5. Joint angles (upper left), joint velocities (middle left), controls (lower left), end-effector trajectory (upper right) and velocity (lower right) for the throwing motion on UR10. The ball trajectory is displayed with dashed green lines on the upper-right plots. Both the velocities, controls and end-effector position satisfy their respective bounds, displayed in red. The minimum target end-effector velocity is also satisfied.

B. Obstacle-avoidance

LQR with obstacles. We extend the exemple of the bound-constrained LQR (37) with path constraints that consist in avoiding obstacles. We consider avoiding the interiors of polyhedral sets of the form $P^{(j)} = \{x \mid C^{(j)}x \leqslant d^{(j)}\}$ which is the piecewise linear constraint

$$\max_i \, (C^{(j)}x - d^{(j)})_i \geqslant 0. \tag{38}$$

These constraints make the problem nonconvex and thus cannot be handled by standard convex solvers. Fig. 6 shows an example with both obstacles and control which saturate.

UR10 – reach task with obstacles. The goal is for the end-effector $p_e(q)$ to reach a target $\bar{p} \in \mathbb{R}^3$, expressed as a terminal cost $\ell_{\mathrm{f}}(x) = \frac{1}{2}\|p_e(q) - \bar{p}\|^2_{W_{\mathrm{ee}}}$. We also impose waypoint constraints at $t_0, t_1 \in (0, T)$, with time horizon $T = 3\,\mathrm{s}$. As obstacles, we choose simple vertical cylinders of radius r_C and impose that they should not collide with spheres of radius r_S centered around given frames p_j (the end-effector and wrist links of the UR10). This condition is expressed using the distance from the sphere center p_j to the cylinder axis: $\|p_j - \mathrm{proj}_{\text{cyl. axis}}(p_j)\| \geqslant r_S + r_C$. Figure 1

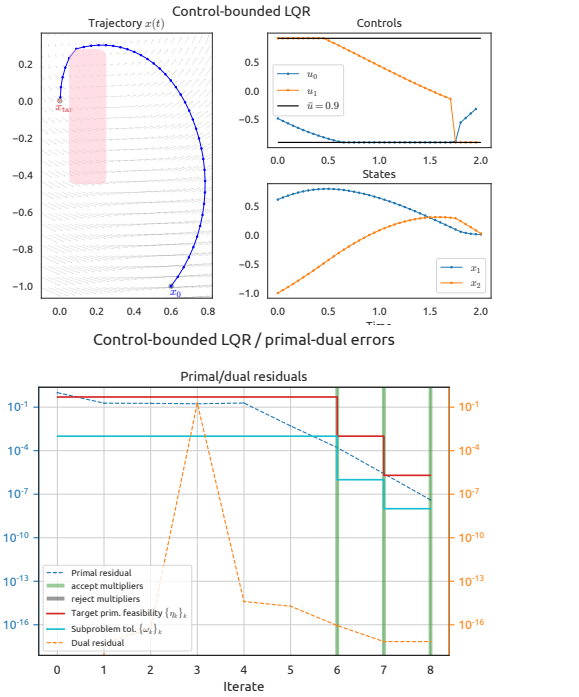

Fig. 6. **Top:** The pink area is a rectangular obstacle defined by (38). The trajectory avoids the area (at the discretization nodes) and the controls saturate: both constraints are satisfied. **Bottom:** Evolution of the primal-dual residuals after each step (backward and forward pass). We obtain very fast convergence in a handful of steps.

illustrates the motion on the UR10 robot, and Fig. 7 controls and velocities.

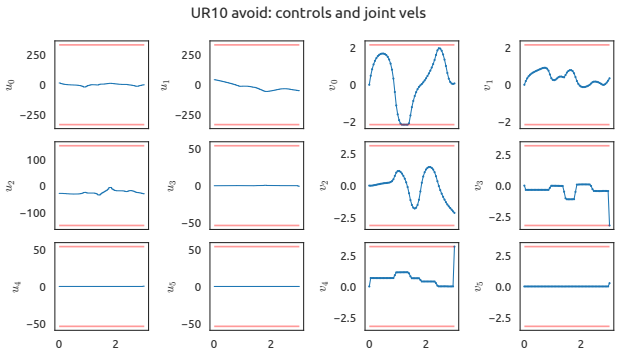

Fig. 7. Controls and velocities for the UR10 reach task. Translucent red lines indicate control and velocity bounds. The velocities saturate only for a few axes in the middle and end of the trajectory (the arm bows down to avoid the obstacles, and lurches forward to reach the final waypoint).

V. Conclusion

In this work, we have introduced a new approach for solving generic NLPs with equality and inequality constraints. We propose combining the BCL globalization strategy [19] with the minimization of a relaxed semi-smooth primal-dual Augmented Lagrangian function inspired by [21]. We then apply this approach to extend the framework of equality-constrained [18] and dynamics-implicit [20] DDP to the case of inequality constraints. It results in an overall second-order quasi-Newton-like algorithm for solving constrained DDP problems. We finally highlight the numerical efficiency of

our method on various sets of standard case-studies from the robotic literature. These contributions pave the way towards more advanced numerical methods for dealing with complex optimization problems in robotics, with the ambition of significantly reducing the computational burden, increase the numerical robustness of the trajectory optimization methods while also lowering the need of manually tuning underlying hyper-parameters. As future work, we plan to implement our contributions in C++ within the Crocoddyl library [40], to properly account for equality and inequality constraints in trajectory optimization.

REFERENCES

[1] J. T. Betts and S. L. Campbell, "Discretize then optimize," *Mathematics for industry: challenges and frontiers*, 2005.

[2] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, 1987.

[3] 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*. Springer, 2006.

[4] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation," in *Nonlinear Model Predictive Control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, vol. 384.

[5] D. M. Murray and S. J. Yakowitz, "Differential dynamic programming and Newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 3, 1984.

[6] 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*, 2012-10-01.

[7] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014-05.

[8] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*, 2017.

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

[10] A. Pavlov, I. Shames, and C. Manzie. Interior Point Differential Dynamic Programming.

[11] S. Singh, J.-J. Slotine, and V. Sindhwani. Optimizing Trajectories with Closed-Loop Dynamic SQP.

[12] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, 2009.

[13] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, 2017, vol. 2.

[14] M. Toussaint, "A novel augmented lagrangian approach for inequalities and convergent any-time non-central updates," *arXiv preprint arXiv:1412.4329*, 2014.

[15] T. A. Howell, B. E. Jackson, and Z. Manchester, "ALTRO: A Fast Solver for Constrained Trajectory Optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019-11.

[16] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou. Constrained Differential Dynamic Programming Revisited.

[17] G. Lantoine and R. Russell, "A Hybrid Differential Dynamic Programming Algorithm for Constrained Optimal Control Problems. Part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, 2013-08-23.

[18] S. Kazdadi, J. Carpentier, and J. Ponce, "Equality Constrained Differential Dynamic Programming," in *2021 IEEE International Conference on Robotics and Automation*, 2021-05-30.

[19] A. Conn, N. Gould, and P. Toint, "A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds," *SIAM Journal on Numerical Analysis*, vol. 28, 1991-04-01.

[20] W. Jallet, N. Mansard, and J. Carpentier, "Implicit Differential Dynamic Programming," in *International Conference on Robotics and Automation (ICRA 2022)*. IEEE Robotics and Automation Society, 2022-05.

[21] P. E. Gill and D. P. Robinson, "A primal-dual augmented Lagrangian," *Computational Optimization and Applications*, vol. 51, no. 1, 2021-04-04.

[22] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research. Springer, 2006.

[23] R. T. Rockafellar, "Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming," *Mathematics of Operations Research*, vol. 1, no. 2, 1976.

[24] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[25] D. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 1982.

[26] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: A Newton-type Proximal Augmented Lagrangian Method for Quadratic Programs," *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2019-12.

[27] ——. QPALM: A Proximal Augmented Lagrangian Method for Nonconvex Quadratic Programs.

[28] D. P. Robinson, "Primal-dual methods for nonlinear optimization," Ph.D. dissertation, UNIVERSITY OF CALIFORNIA, SAN DIEGO, 2007.

[29] A. R. Conn, N. I. M. Gould, and P. L. Toint, "A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds," *SIAM Journal on Numerical Analysis*, vol. 28, no. 2, 1991.

[30] ——, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*. Springer Science & Business Media, 2013.

[31] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017-09.

[32] A. De Marchi, "On a primal-dual Newton proximal method for convex quadratic programs," *Computational Optimization and Applications*, vol. 81, no. 2, 2022-03.

[33] M. Hintermüller, "Semismooth newton methods and applications," *Department of Mathematics, Humboldt-University of Berlin*, 2010.

[34] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, 2020.

[35] S. Brossette, A. Escande, and A. Kheddar, "Multicontact postures computation on manifolds," *IEEE Transactions on Robotics*, vol. 34, no. 5, 2018.

[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, 2006-03.

[37] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[38] J. Carpentier and N. Mansard, "Analytical Derivatives of Rigid Body Dynamics Algorithms," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, 2018-06-26.

[39] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, 2020-09.

[40] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020.