Implicit Differential Dynamic Programming

Wilson Jallet, Nicolas Mansard, Justin Carpentier

Implicit Differential Dynamic Programming

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

***Abstract*— Over the past decade, the Differential Dynamic Programming (DDP) method has gained in maturity and popularity within the robotics community. Several recent contributions have led to the integration of constraints within the original DDP formulation, hence enlarging its domain of application while making it a strong and easy-to-implement competitor against alternative methods of the state of the art such as collocation or multiple-shooting approaches. Yet, and similarly to its competitors, DDP remains unable to cope with high-dimensional dynamics within a receding horizon fashion, such as in the case of online generation of athletic motion on humanoid robots. In this paper, we propose to make a step toward this objective by reformulating classic DDP as an implicit optimal control problem, allowing the use of more advanced integration schemes such as implicit or variational integrators. To that end, we introduce a primal-dual proximal Lagrangian approach capable of handling dynamic and path constraints in a unified manner, while taking advantage of the time sparsity inherent to optimal control problems. We show that his reformulation enables us to relax the dynamics along the optimization process by solving it inexactly: far from the optimality conditions, the dynamics are only partially fulfilled, but continuously enforced as the solver get closer to the local optimal solution. This inexactness enables our approach to robustly handle large time steps (100 ms or more), unlike other DDP solvers of the state of the art, as experimentally validated through different robotic scenarios.**

I. Introduction

Optimal control is a very convenient way to formally and practically compute the movement that a robot has to achieve [1], [2], [3], [4]. Among the various possibilities to numerically solve an optimal control problem, the differential dynamic programming [5], [6] algorithm provides several advantages that makes it a frequent choice to solve optimization trajectory problems. First, it is optimally exploiting the sparsity inherently obtained from the temporal structure of the trajectory problem. Second, as a shooting method, DDP ensures to fulfill the robot dynamics at all time. Yet, a major drawback of DDP is its inability to handle constraints in its vanilla formulation.

Recent works have attempted to account for these constraints using various nonlinear programming strategies for solving constrained optimization problems [7] such as penalty [6], [8], [9], augmented Lagrangian [10], [11], [12], [13], sequential quadratic programming [14], [15], [16] or interior-point methods [17]. All these works have in common addressing various types of robot constraints, but none of them have considered the case of implicit integrators, where the system dynamics are implicitly defined by the zero of a given function. Implicit integrators lead to more robust integration schemes, with better theoretical guarantees in terms of stability and energy preservation, while potentially being able to operate on larger integration steps [18]. It is expected from other fields of optimal control that formulating a solver based on such integrator would boost the capabilities of optimization [19], in particular by enabling larger integration intervals, hence reducing the number of integration nodes (hence number of variables) for a given horizon or enlarging the preview horizon for a similar computational cost. Among these implicit schemes, variational integrators might be formulated [20], with the nice guarantee of keeping the exact energy level of the system.

In this paper, we introduce an implicit DDP (IDDP) formulation able to work with such advanced integrators. We formulate a new backward pass which also backpropagates a series of gains corresponding to the Lagrange multipliers. We then propose a proximal augmented Lagrangian algorithm to properly handle the constraints while ensuring improved numerical stability. This algorithm is capable of handling dynamic and path constraints in a unified manner, while taking advantage through the proposed backward pass of the time sparsity inherent to optimal control problems. We then propose a complete implementation of this idea based on the rigid-body dynamics library Pinocchio [21] and the automatic differentiation framework Casadi [22], and evaluate it on several robotic models of various complexity, in comparison with classic DDP implementation [9]. The benchmarks demonstrate good properties of the solver (globalization, convergence rate, accuracy at convergence) and of the variational integrator that we implemented.

The paper is organized as follows: we first introduce the implicit DDP formulation and its proximal algorithm (Sec. II), recall the notion of variational integrators and depict its inclusion within IDDP (Sec. III). Finally, we report the results with our implementation and their comparison against Crocoddyl (Sec. IV).

II. Implicit differential dynamic programming

This section introduces the notion of implicit differential dynamic programming, which consists in reformulating differential dynamic programming by considering the dynamics as implicit constraints of the problem. Rooted on [13], we also introduce a primal-dual proximal Lagrangian approach

[a] INRIA and [b] Département d'informatique de l'ENS, Paris, France
[c] LAAS-CNRS, 7 Avenue du Colonel Roche, F-31400 Toulouse, France
*corresponding author: wilson.jallet@inria.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).

capable of handling dynamical and path constraints in a unified manner, while taking advantage of the temporal sparsity inherent to optimal control problems.

A. Problem formulation and transcription

Problem formulation. In this work, we focus on equality-constrained optimal control problems of the form:

$$\min_{\mathbf{x},\mathbf{u}} \quad \int_0^T \ell(t, x(t), u(t))\, dt + h(x(T)) \tag{1a}$$
$$\dot{x}(t) = f(t, x(t), u(t)) \tag{1b}$$
$$(x(t), u(t)) \in \mathcal{C}(t) \quad \forall t \in [0, T[\tag{1c}$$
$$x(0) = \bar{x}_0, x(T) \in \mathcal{C}(T), \tag{1d}$$

where f is the continuous system dynamics and $\bar{x}_0$ is the initial state condition. We denote $\mathcal{X}$ and $\mathcal{U}$ denote the state and control sets, respectively. We assume that the sets of *path constraints* $\mathcal{C}(t)$ can be expressed as equality constraints $\{(x(t), u(t) \in \mathcal{X} \times \mathcal{U} : g(x(t), u(t)) = 0\}$. The extension to inequality constrained problems is left as a future work. The state space $\mathcal{X}$ is in general a Lie group; we denote its addition (integration) operation $\oplus$ and its difference (retraction) operation as $\ominus$.

Explicit transcription. As classically done within the numerical optimal control literature [1], [19], problem (1) can be transcribed into a finite dimensional nonlinear programming problem of the form:

$$\min_{\mathbf{x},\mathbf{u}} \quad \sum_{t=0}^{N-1} \ell_t(x_t, u_t) + h_T(x_N) \tag{2a}$$
$$\text{s.t.} \quad x_{t+1} = f_t(x_t, u_t),\ t = 0, \dots, N-1$$
$$g_t(x_t, u_t) = 0 \tag{2b}$$
$$x_0 = \bar{x}_0 \in \mathcal{X},$$

where $\mathbf{x} \overset{\text{def}}{=} (x_0, \dots, x_N)$ and $\mathbf{u} \overset{\text{def}}{=} (u_0, \dots, u_{N-1})$. The control trajectory $u(.)$ has been discretized on a given time grid and f_t is the discrete dynamics obtained from f using an explicit or implicit discrete integrator.

Solving such an equality constrained problem can be performed using classical methods of nonlinear programming [7] while using sparse linear algebra routines [23], or either by unrolling the time induced sparsity by exploiting constrained differential dynamic approaches [16], [15], [12], [13], [17] or even simply by accounting constraints at the dynamic level [24]. All these methods require the dynamic equations to be perfectly solved, following a bi-level programming strategy. While explicit dynamics can be efficiently evaluated with high numerical accuracy [25], implicit dynamics typically require several steps of the Newton-Raphson algorithm in order to accurately compute x_{t+1}, as done in [26], which in turn may slow down the whole solving process.

Implicit transcription. Yet, solving the dynamics with a high accuracy is not required to make the solver converge to an optimal solution, as suggested by inexact methods within the optimization and machine learning community [27]. In this work, we propose to "relax" the dynamics and solve them inexactly by considering them as standard constraints, leading to the following implicit transcription scheme:

$$\min_{\mathbf{x},\mathbf{u}} \sum_{t=0}^{N-1} \ell_t(x_t, u_t) + h_T(x_N) \tag{3a}$$
$$f_t(x_t, u_t, x_{t+1}) = 0,\ t = 0, \dots, N-1 \tag{3b}$$
$$g_t(x_t, u_t) = 0 \tag{3c}$$
$$x_0 = \bar{x}_0. \tag{3d}$$

Contrary to standard constraints such as (3c), Eq. (3b) depends on both the current state x_t and of the next state x_{t+1} at a given time instant t, which leads to an alternative formulation of the dynamic recursion, as shown hereafter.

By denoting $c(\mathbf{x}, \mathbf{u})$ the entire stack of constraints, the Lagrangian function associated with (3) is given by:

$$\mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) = \sum_{i=0}^{N-1} \ell(x_i, u_i) + h(x_N) + \boldsymbol{\lambda}^\top c(\mathbf{x}, \mathbf{u}), \tag{4}$$

where $\boldsymbol{\lambda}$ is the stack of Lagrangian multipliers related to the path and dynamic constraints, also known as co-state in optimal control [28]. The Karush-Kuhn-Tucker (KKT) [29] conditions for optimality of (3) are thus given by:

$$\begin{aligned} \nabla_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) &= 0 \\ x_0 &= \bar{x}_0 \\ g_t(x_t, u_t) = 0,\ f_t(x_t, u_t, x_{t+1}) &= 0, \quad 0 \leq t < N. \end{aligned} \tag{5}$$

B. Dynamic programming of the implicit transcription

The dynamic programming recursion related to (3) is

$$\begin{aligned} V_t(x) &= \min_{u,y} \ell_t(x, u) + V_{t+1}(y) \\ &\text{s.t. } f_t(x, u, y) = 0 \end{aligned} \tag{6}$$

with V_t the value function at time t and $V_N(x) = h(x)$. Without loss of generality and to simplify the presentation, we consider $g_t(x, u)$ included within $f_t(x, u, y)$. Using the same ideas behind DDP [30], [6], we can iteratively solve (3) using successive second-order approximations of the problem and dynamic programming. The approach combines standard sequential quadratic programming (SQP) from the optimization literature [7] with dynamic programming.

We introduce the following Hamiltonian function:

$$Q_t(x, u, y, \lambda) = \ell_t(x, u) + \lambda^\top f_t(x, u, y) + V_{t+1}(y). \tag{7}$$

Backward pass. In the sequel, we use the usual notations to denote the partial derivatives of Q (Q_x, Q_{xu}, etc.). In the backward pass, the method solves a sequence of QP subproblems for $t = N-1, \dots, 0$, following the recursion:

$$\bar{V}_t(\delta x) = \min_{\delta u, \delta y} \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \\ \delta y \end{bmatrix}^\top \begin{bmatrix} Q_{xx} & Q_{xu} & Q_{xy} \\ Q_{ux} & Q_{uu} & Q_{uy} \\ Q_{yx} & Q_{yu} & Q_{yy} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta y \end{bmatrix} \tag{8a}$$
$$+ Q_x^\top \delta x + Q_u^\top \delta u + Q_y^\top \delta y + q$$
$$\text{s.t. } f + F_x \delta x + F_u \delta u + F_y \delta y = 0 \tag{8b}$$

where $f = f_t(x, u, x_y)$, and back-propagates a sequence of model value functions $\bar{V}_t$ which are quadratic functions of the states x_t,

$$\bar{V}_t(x + \delta x) = v_t + V_x^\top \delta x + \frac{1}{2}\delta x^\top V_{xx} \delta x.$$

The partial derivatives of the Hamiltonian Q are given by

$$\begin{aligned} Q_x &= \ell_x + F_x^\top \lambda & Q_u &= \ell_u + F_u^\top \lambda \\ Q_y &= V_x' + F_y^\top \lambda & Q_{xx} &= \ell_{xx} + \lambda \cdot F_{xx} \\ Q_{xu} &= \ell_{xu} + \lambda \cdot F_{xu} & Q_{xy} &= \lambda \cdot F_{xy} \\ Q_{uu} &= \ell_{uu} + \lambda \cdot F_{uu} & Q_{uy} &= \lambda \cdot F_{uy} \\ Q_{yy} &= V_{yy}' + \lambda \cdot F_{yy} & q &= \ell_t + \lambda^\top f + v' \end{aligned} \tag{9}$$

and $\bar{V}'$ is a shorthand for the next-timestep model value function $\bar{V}_{t+1}$ (so $V_x' = \nabla_x \bar{V}_{t+1}$).

The KKT conditions of the underlying QP are

$$\underbrace{\begin{bmatrix} Q_{uu} & Q_{uy} & F_u^\top \\ Q_{yu} & Q_{yy} & F_y^\top \\ F_u & F_y & 0_{n_c} \end{bmatrix}}_{=\mathcal{K}} \begin{bmatrix} \delta u \\ \delta y \\ \delta \lambda \end{bmatrix} = - \begin{bmatrix} Q_u + Q_{ux}\delta x \\ Q_y + Q_{yx}\delta x \\ f + F_x \delta x \end{bmatrix}. \tag{10}$$

The solution of this equation depends on the value of the step δx. Assuming the left-hand side KKT matrix $\mathcal{K}$ is invertible, we have that we can write

$$\delta u = k + \mathbf{K}\delta x,\ \delta y = a + \mathbf{A}\delta x,\ \delta \lambda = \xi + \mathbf{\Xi}\delta x, \tag{11}$$

where the zeroth- and first-order sensitivities (k, a, ξ) and $(\mathbf{K}, \mathbf{A}, \mathbf{\Xi})$ are solutions of the matrix equation

$$\mathcal{K} \begin{bmatrix} k & \mathbf{K} \\ a & \mathbf{A} \\ \xi & \mathbf{\Xi} \end{bmatrix} = - \begin{bmatrix} Q_u & Q_{ux} \\ Q_y & Q_{yx} \\ f & F_x \end{bmatrix}. \tag{12}$$

Forward pass, linear rollout. In contrast to usual DDP approaches [30], [6], [12], [13], we do *not* perform a nonlinear rollout but a linear one, which corresponds to taking a primal-dual step in SQP methods [7].The linear increments are obtained by the recursion:

$$\delta x_{t+1} = a_t + \mathbf{A}\delta x_t,\ \delta u_t = k_t + \mathbf{K}\delta x_t,\ \delta \lambda_{t+1} = \xi_t + \mathbf{\Xi}\delta x_t. \tag{13}$$

Similarly to [10], [13], (13) depicts an affine update of the multipliers $\delta\lambda_{t+1}$. The next iterates are given by a partial step α

$$\mathbf{x}^+ = \mathbf{x} \oplus \alpha\delta\mathbf{x},\ \mathbf{u}^+ = \mathbf{u} + \alpha\delta\mathbf{u},\ \boldsymbol{\lambda}^+ = \boldsymbol{\lambda} + \alpha\delta\boldsymbol{\lambda},$$

where $\alpha \in (0, 1]$ is determined by a line-search procedure, which will be expanded upon in a latter section.

C. A proximal-point reformulation

The so-called KKT matrix $\mathcal{K}$ introduced in (10) is not guaranteed to be nonsingular, and might have very poor conditioning. Some authors in the optimization literature [31] advocate for regularization of the matrix in SQP. This idea is close to that of the method of multipliers [7], which optimizes a penalized, augmented Lagrangian function. We propose introducing an inexact proximal point method in the dual variables, where we update the primal and dual variables (states $\mathbf{x}$, controls $\mathbf{u}$ and multipliers $\boldsymbol{\lambda}$) as follows:

$$(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}, \boldsymbol{\lambda}^{k+1}) \approx \arg\min_{\mathbf{x},\mathbf{u}} \max_{\boldsymbol{\lambda}} \mathcal{L}_{\mu_k}^{\text{prox}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^k), \tag{14}$$

with:

$$\mathcal{L}_{\mu_k}^{\text{prox}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^k) = \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) - \frac{1}{2\mu_k}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^k\|_2^2. \tag{15}$$

The exact update is known to be equivalent to the method of multipliers (see [32]). The proximal iteration above is also equivalent to minimizing the augmented Lagrangian function

$$\mathcal{L}_{\mu_k}^{\text{AL}}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}^k) = \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) + \frac{\mu_k}{2}\|c(\mathbf{x}, \mathbf{u})\|_2^2, \tag{16}$$

but leading to more stable numerical schemes, as recalled in [13].

The new KKT equations given by:

$$\underbrace{\begin{bmatrix} Q_{uu} & Q_{uy} & F_u^\top \\ Q_{yu} & Q_{yy} & F_y^\top \\ F_u & F_y & -\frac{1}{\mu_k} I_{n_c} \end{bmatrix}}_{\mathcal{K}_{\mu_k}} \begin{bmatrix} a & \mathbf{A} \\ k & \mathbf{K} \\ \xi & \mathbf{\Xi} \end{bmatrix} = - \begin{bmatrix} Q_u & Q_{ux} \\ Q_y & Q_{yx} \\ f - \frac{\lambda - \lambda^k}{\mu_k} & F_x \end{bmatrix} \tag{17}$$

are very similar to the previous ones depicted in (12). Contrary to (12), the conditioning of the matrix is improved thanks to $-\frac{1}{\mu_k} I_{n_c}$. The benefit of the proximal reformulation of the Lagrangian, as shown in [33], [13], enables our approach to cope with ill-posed problems, where F_u and F_y are potentially rank-defficient.

Globalization and inexactness strategy. The inexactness of the iterates can be controlled using strategies such as the bound-constrained Lagrangian (BCL) [34], as advocated by [11], [13]. Additionally, the BCL strategy allows to automatically adjust μ_k according to the primal/dual feasibility of the problem given by (5) and to schedule the tolerance ϵ_k of the inner loop, related to the dual feasibility and the tolerance of the outer loop $\eta^\star$, related to the overall primal feasibility.

Stopping criterion. We distinguish the inner iterations (the minimization phase where we find the inexact proximal iterate) from the outer iterations.

For the inner iterations, we can consider two different criteria: (i) the gradient of the augmented Lagrangian:

$$\|\nabla_{\mathbf{x},\mathbf{u}} \mathcal{L}_{\mu_k}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}^k)\|_\infty \leq \epsilon_k \tag{18}$$

or (ii) the primal-dual residuals of the proximal saddle-point:

$$\left\| \begin{bmatrix} \nabla_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) \\ c(\mathbf{x}, \mathbf{u}) - \frac{1}{\mu_k}(\boldsymbol{\lambda} - \boldsymbol{\lambda}^k) \end{bmatrix} \right\|_\infty \leq \epsilon_k, \tag{19}$$

with ω_k the tolerance of the inner iterations at iteration k.

For the outer iterations, we can consider the distance between consecutive proximal iterates $\boldsymbol{\lambda}^k$ and $\boldsymbol{\lambda}^{k+1}$, which are strongly correlated to the gradient of the so-called Moreau envelope [35]. Indeed, the Moreau envelope corresponds to the value of the saddle-point (14) defined above:

$$\mathscr{M}_{\mu_k}(\boldsymbol{\lambda}^k) \triangleq \max_{\boldsymbol{\lambda}} \min_{\mathbf{x},\mathbf{u}} \mathcal{L}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) - \frac{1}{2\mu_k}\|\boldsymbol{\lambda} - \boldsymbol{\lambda}^k\|_2^2. \tag{20}$$

and its gradient corresponds to:

$$\nabla_{\boldsymbol{\lambda}} \mathscr{M}_{\mu_k}(\boldsymbol{\lambda}^{k+1}) = c(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) = -\frac{1}{\mu_k}(\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k).$$

This means that a stopping criterion on proximal dual iterates can be translated into a criterion on the constraints' norm:

$$\|c(\mathbf{x}, \mathbf{u})\|_\infty \leq \eta^\star. \tag{21}$$

Primal proximal regularization. In addition to regularizing the dual variables, it is possible to enforce strong convexity in the primal variables by adding corresponding proximal terms:

$$\min_{\mathbf{x},\mathbf{u}} \max_{\boldsymbol{\lambda}} \mathcal{L}^{\text{prox}}_{\mu_k}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}) + \frac{\rho_k}{2}\|\mathbf{x} \ominus \mathbf{x}^k\|_2^2 + \frac{\rho_k}{2}\|\mathbf{u} - \mathbf{u}^k\|_2^2, \tag{22}$$

By doing so, the resulting KKT matrix $\mathcal{K} + \text{diag}(\rho_k I, \rho_k I, -\frac{1}{\mu_k} I)$ depicts a better inertia [36], enforcing the overall numerical conditioning of the problem while making our formulation robust against singular cases as depicted in the results section IV.

Linesearch. We use the augmented Lagrangian as a merit function:

$$\min_{\alpha} \mathcal{L}^{\text{AL}}_{\mu_k}(\mathbf{x} \oplus \alpha\delta\mathbf{x}, \mathbf{u} + \alpha\delta\mathbf{u}, \boldsymbol{\lambda}^k). \tag{23}$$

An alternative is to use the primal-dual augmented Lagrangian function [36], which accounts for steps in the inner multiplier λ. We use a backtracking Armijo linesearch as in [6], [9], although more sophisticated approaches involving e.g. quadratic interpolation could be used.

III. Advanced implicit integrators

In this section, we review the concept of variational integrators and show they can be easily plugged within the implicit differential dynamic setting.

A. Variational integrators

Variational integrators [37], [20] are a class of numerical integrators for second-order dynamical systems derived from variational principles in physics, which are known to exhibit better energy conservation properties when compared to classical numerical integration methods such as Euler or Runge-Kutta schemes over Lagrangian equations of motion [20].

Continuous formulation. Variational integrators (VI) rely on the discretization of the so-called least action principle [37], [20]. Given a dynamical system's Lagrangian function

$$L(q, \dot{q}) = \underbrace{T(q, \dot{q})}_{\text{kinetic energy}} - \underbrace{V(q)}_{\text{potential energy}}, \tag{24}$$

the least action principle states that the state trajectory $q : [0, T] \to \mathcal{Q}$ is a *stationary* point of the action functional defined by:

$$\mathcal{S}(q) \triangleq \int_0^T L(q, \dot{q})\, dt. \tag{25}$$

In continuous time, the stationary conditions lead to the classical Euler-Lagrange equations

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = 0, \tag{26}$$

from which the so-called Lagrangian equations of motion can be derived, resulting in:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = 0, \tag{27}$$

where M, C and g stands for the mass matrix, Coriolis and centrifugal effects and the generalized gravity associated to the kinetic and potential energy. $\dot{q}$ and $\ddot{q}$ are the first and second order time derivatives of q respectively.

Discrete formulation. Direct discretization of the variational principle introduces a discrete action sum

$$S^d(\mathbf{q}) = \sum_{i=0}^{N-1} L_d(q_i, q_{i+1}), \tag{28}$$

where $L_d(q_i, q_{i+1})$ is a term for the quadrature of the action integral (25), and $\mathbf{q} = (q_0, \ldots, q_N)$ are the generalized coordinates at the different discrete time instants. We chose a good empirical quadrature, the trapezoidal rule. We obtain discrete dynamics by looking at the stationary conditions $\nabla_{\mathbf{q}} S^d(\mathbf{q}) = 0$. These conditions lead to new discrete dynamics, the so-called *Discrete Euler-Lagrange equation* (DEL):

$$\partial_2 L_d(q_{i-1}, q_i) + \partial_1 L_d(q_i, q_{i+1}) = 0. \tag{29}$$

The VI is a two-step integrator, where we recover (q_i, q_{i+1}) from (q_{i-1}, q_i).

Solving the forward dynamics can be done using Newton-Raphson iterations, as advocated by [20]. However, for e.g. stiff systems, care has to be taken to scale these Newton steps as to ensure convergence.

Control and constrained dynamics. Forces not included inside of the system's Lagrangian function L, e.g. control, friction and contact forces, can be included into the variational principle through their *virtual work*; the corresponding variational principle is called the virtual work principle. In continuous time, for a given set of external forces $f(t)$, this reads

$$d\mathcal{S}(\mathbf{q}) \cdot \delta\mathbf{q} + \int_0^T f(t) \cdot \delta q(t)\, dt = 0. \tag{30}$$

The corresponding Euler equation derived from the stationary conditions yield the usual differential equation

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = f(t).$$

This also works for external forces, which are given by $f_c(t) = J_c(q)^\top \lambda(t)$ with J_c the Jacobian related to the external force. In discrete-time, the complete dynamical equation including bilateral constraints reads

$$\begin{aligned} \partial_2 L_d(q_0, q_1) + \partial_1 L_d(q_1, q_2) + u\Delta t + D\phi_0(q_0)^\top \lambda_1 = 0 \\ \phi_1(q_1) = 0. \end{aligned} \tag{31}$$

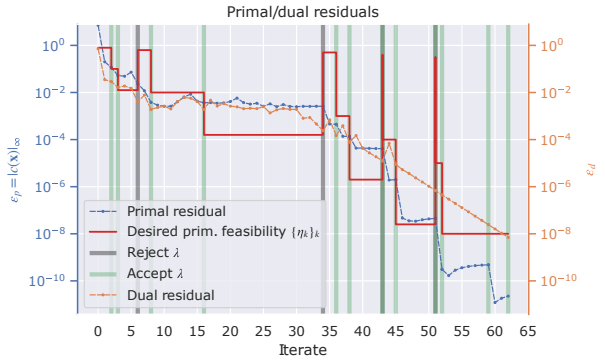

Fig. 1. Convergence plot for the Acrobot, with a time step of $\Delta t = 0.1$.

For control, the augmented state variable is given by configuration, velocity, and contact forces $z = (q, v, \lambda)$. The system of equations (31) can then be plugged in (3) in order to replace (3b).

IV. Results

In this section, we illustrate and analyse the performances of our implicit differential dynamic programming solution over several robot settings, namely the control of an underactuated acrobot, a pick-and-place scenario with the UR-5 robot and finally motions on Solo-8 and Solo-12 quadrupedal platforms [38]. We notably show the benefits of our implicit formulation against Crocoddyl [9], an alternative explicit DDP solver of the state of the art, which is able to operate with Semi-Explicite and Runge-Kutta 4 integration schemes.

Our framework has been developed in Python and leverages the versatile CasADi [22] autodiff framework in conjunction with Pinocchio [21], [39] to evaluate the variational integrators and their derivatives, while relying on the analytical derivatives of Rigid Body Dynamics algorithms [40] for the other types of integrators. We will make our framework freely available upon acceptance. More details on these experiments, in particular the resulting motions, are also reported in the companion video.

Acrobot. The acrobot is a simple underactuated 2DoF system. The task we solve is to get the pendulum upright, whilst penalizing the cumulative control torque. We formulate the task as an hard constraint $q = 0$. We also impose terminal velocity $\dot{q} = 0$, which leads to an over-constrained problem. We use the variational integrator to integrate over a large time step $\Delta t = 0.1$s. Figure 1 shows the convergence behaviour (in both primal and dual residuals) of our solver.

UR5 robotic arm. We impose a terminal position and null velocity of the end-effector of the UR5, as well as the end-effector passing through given waypoints p_0, p_1 at time instants $0 \leq t_0 < t_1 < T$. This is formulated as a hard constraint. We test the problem with random waypoint locations on a number of seeds, with multiple integration time steps Δt. Fig. 3 illustrates the motion obtained using our method.

The implementation of standard DDP we use is from the Crocoddyl library [9]; constraints are included as soft

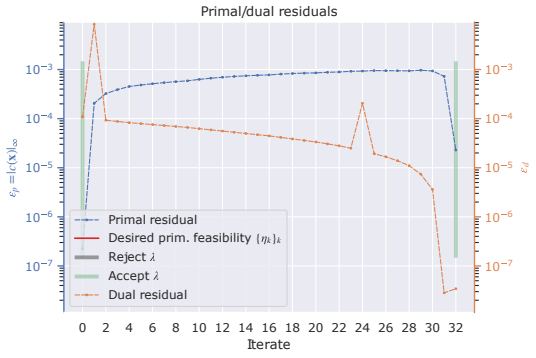

Fig. 2. Convergence of the implicit solver for the "bowing" movement.

quadratic penalties

$$w_g \| p(q(t_i)) - p_i \|_2^2 \tag{32}$$

with $w_g = 600$. Note there is no guarantee that the waypoint constraints are satisfied at convergence.

For our experiments, we pick waypoints randomly from the workspace with 10 fixed seeds, defining 10 different instances of the task. We solve the task using i) our solver with the semi-implicit Euler method, ii) with the variational integrator, as well as iii) DDP with semi-implicit Euler iv) DDP with RK4. We set a maximum number of iterations of 500. We also consider the residual between the instantaneous mechanical power dE_{m}/dt and the control instantaneous power $P = \langle u(t), \dot{q} \rangle$, which would be zero for an ideal integrator of the system, by the work-energy principle.

TABLE I

UR5: Proportion of solved instances over 10 runs.

Method	integrator	10 ms	20 ms	50 ms	100 ms
IDDP (ours)	Euler	1.0	1.0	1.0	1.0
IDDP	Variational	1.0	1.0	1.0	1.0
Croco	Euler	1.0	1.0	1.0	0.8
Croco	RK4	0.7	0.6	0.6	0.3

TABLE II

Optimal cost for a pick-and-place task with UR5.

Method	integrator	10 ms	20 ms	50 ms	100 ms
IDDP	Euler	1.197	1.197	1.200	1.221
IDDP	Variational	1.219	1.213	1.196	1.181
Croco	Euler	1.748	1.335	6.017	1.183
Croco	RK4	N/A	N/A	N/A	N/A

TABLE III

Average (standard dev.) of residual energy $E_m - W$ over all instances.

Method	integrator	10 ms	20 ms	50 ms	100 ms
IDDP	Euler	5.81 (3.35)	5.44 (2.91)	5.21 (2.92)	4.54 (2.82)
IDDP	Variational	5.35 (2.89)	5.27 (2.86)	4.63 (2.64)	3.09 (1.88)
Croco	Euler	10.9 (5.37)	9.86 (6.09)	12.7 (14.5)	4.0e42 (1.3e43)
Croco	RK4	N/A	N/A	N/A	N/A

Convergence. Table I shows the proportion of scenarios for which the solvers converged, depending on the choice of integrator and time step. Our method (denoted IDDP) converges on every instance.

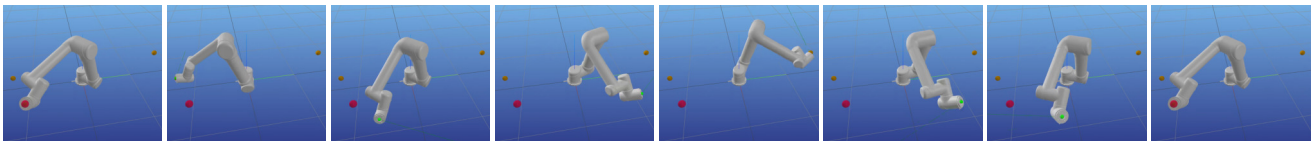

Fig. 3. Optimal motion for the UR5 pick-and-place task.

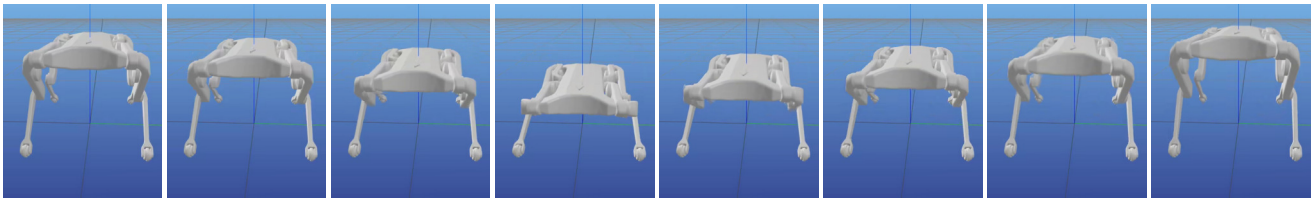

Fig. 4. "Bowing" movement on Solo-8.

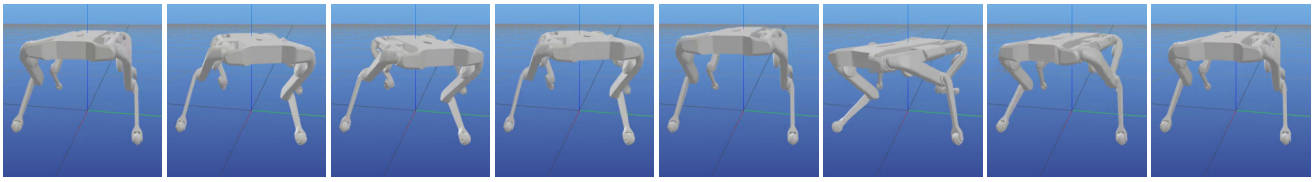

Fig. 5. "Spin" movement on Solo-12.

Globalisation. Table II shows that our method achieves better trajectory cost across multiple value of integration time step Δt, than the reference implementation of DDP. The table was created using results from a single instance of the pick-and-place task. Results for the RK4 integrator are not available, since the reference implementation of DDP failed for this instance.

Energy. Table III shows the average and standard deviation of the residual between the variation of mechanical energy and work $\Delta E_{\mathrm{m}} - W$ over the 10 scenarios. Across the different time integration steps, the implicit DDP algorithm displays a smaller residual, especially when used with the variational integrator. As for semi-implicit Euler, our solver still achieves smaller residuals: the trajectories produced by DDP actually show larger amplitudes of movement, which exacerbates energy defects in the integrator.

Solo quadruped. The Solo robot is an example of under-actuated system subject to contact forces. We formulate dynamics using the constrained variational integrator (31): this integrator removes the need for contact linearization and baumgarte constraint corrections, as in [9]. We impose the constraints as $p_j(q) = \bar{p}_j \in \mathbb{R}^3$ where the $\bar{p}_j$ are points in the $z = 0$ plane and p_j are feet locations. Note that for the Solo-8 robot, which has fewer degrees of freedom, and whose feet cannot move laterally, the dynamics become over-constrained. We test two tasks, wherein the robot base has to move down (bowing task, Fig. 4) or rotate from one side to the other (Fig. 5). Fig. 2 shows convergence of the solver for a bowing task. The solver converges in about 30 iterations. Primal infeasibility $\|c(\mathbf{x}, \mathbf{u})\|_\infty$ does not improve until the last two iterates, as the penalty parameter μ_0 (initialized at 10) is too low. Yet convergence is achieved quickly.

V. Conclusion

We have proposed a reformulation of the classical DDP to handle dynamical systems modeled under an implicit form. There are several positive consequences to this proposition. First, it immediately unlocks the use of more powerful integrators, such as variational integrators, which better and more stably represent the robot dynamics. Second, we can model systems subject to mechanical constraints (such as legged robots or closed-loop mechanisms) at no extra effort. A benefit is that we can reduce the number of shooting intervals without decreasing the quality and the stability of the optimization, which in turn lowers the total computational cost. While other optimal control solvers were already able to handle implicit integrators, DDP is optimal in its way to handle the sparsity of control problems.

A proximal Lagrangian algorithm is used to solve the underpinning constrained optimization problem. It boils down to a reformulation of the classical DDP backward pass, which leads to improved numerical stability. The new backward pass generates extra gains corresponding to the evaluation of Lagrange multipliers, which opens a route to new feedback terms. Our implementation, which will be open-sourced should the paper be accepted, has been used to empirically evaluate the performance of our approach in several scenarios of various complexities. We have benchmarked the capabilities of our solver against Crocoddyl, a state-of-the-art DDP solver used on several legged and aerial robots, and show that our method converges quickly, more reliably and to a better optimum. We have also demonstrated the interest of variational integrators for handling the simulation of polyarticulated systems.

REFERENCES

[1] 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, pp. 65–93.

[2] G. Schultz and K. Mombaur, "Modeling and optimal control of human-like running," *IEEE/ASME Transactions on mechatronics*, vol. 15, no. 5, pp. 783–792, 2009.

[3] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3346–3351.

[4] J. Carpentier and P.-B. Wieber, "Recent progress in legged robots locomotion control," *Current Robotics Reports*, pp. 1–8, 2021.

[5] D. Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, Jan. 1966, publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207176608921369. [Online]. Available: https://doi.org/10.1080/00207176608921369

[6] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," Oct. 2012, pp. 4906–4913.

[7] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed., ser. Springer series in operations research. New York: Springer, 2006, oCLC: ocm68629100.

[8] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[9] 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," *arXiv:1909.04947 [cs, math]*, Mar. 2020, arXiv: 1909.04947. [Online]. Available: http://arxiv.org/abs/1909.04947

[10] G. Lantoine and R. P. Russell, "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 382–417, 2012.

[11] 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, pp. 5674–5680.

[12] 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)*. Macau, China: IEEE, Nov. 2019, pp. 7674–7679. [Online]. Available: https://ieeexplore.ieee.org/document/8967788/

[13] S. Kazdadi, J. Carpentier, and J. Ponce, "Equality Constrained Differential Dynamic Programming," May 2021. [Online]. Available: https://hal.inria.fr/hal-03184203

[14] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pp. 1168–1175. [Online]. Available: http://ieeexplore.ieee.org/document/6907001/

[15] M. Giftthaler and J. Buchli, "A Projection Approach to Equality Constrained Iterative Linear Quadratic Optimal Control," *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 61–66, Nov. 2017, arXiv: 1805.09403. [Online]. Available: http://arxiv.org/abs/1805.09403

[16] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.

[17] A. Pavlov, I. Shames, and C. Manzie, "Interior point differential dynamic programming," *IEEE Transactions on Control Systems Technology*, 2021.

[18] S. Rubrecht, V. Padois, P. Bidaud, and M. de Broissia, "Constraints Compliant Control: Constraints compatibility and the displaced configuration approach," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei: IEEE, Oct. 2010, pp. 677–684. [Online]. Available: http://ieeexplore.ieee.org/document/5650793/

[19] S. Gros and M. Diehl, "Numerical optimal control," 2019.

[20] E. R. Johnson and T. D. Murphey, "Scalable Variational Integrators for Constrained Mechanical Systems in Generalized Coordinates," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1249–1261, Dec. 2009.

[21] 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.

[22] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[23] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*. Oxford University Press, 2017.

[24] R. Budhiraja, J. Carpentier, C. Mastalli, and N. Mansard, "Differential Dynamic Programming for Multi-Phase Rigid Contact Dynamics," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. Beijing, China: IEEE, Nov. 2018, pp. 1–9. [Online]. Available: https://ieeexplore.ieee.org/document/8624925/

[25] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[26] I. Chatzinikolaidis and Z. Li, "Trajectory optimization for contact-rich motions using implicit differential dynamic programming," *arXiv:2101.08246 [cs, eess]*, Jan. 2021, arXiv: 2101.08246. [Online]. Available: http://arxiv.org/abs/2101.08246

[27] M. Schmidt, N. L. Roux, and F. Bach, "Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization," *arXiv:1109.2415 [cs, math]*, Dec. 2011, arXiv: 1109.2415. [Online]. Available: http://arxiv.org/abs/1109.2415

[28] D. Liberzon, *Calculus of variations and optimal control theory*. Princeton university press, 2011.

[29] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike-20&path=ASIN/0521833787

[30] D. Jacobson, "Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games," *IEEE Transactions on Automatic Control*, vol. 18, no. 2, pp. 124–131, Apr. 1973. [Online]. Available: http://ieeexplore.ieee.org/document/1100265/

[31] S. J. Wright, "Superlinear Convergence of a Stabilized SQP Method to a Degenerate Solution," p. 23, 1998.

[32] R. T. Rockafellar, "Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 97–116, 1976, publisher: INFORMS. [Online]. Available: https://www.jstor.org/stable/3689277

[33] J. Carpentier, R. Budhiraja, and N. Mansard, "Proximal and sparse resolution of constrained dynamic equations," in *Robotics: Science and Systems 2021*, 2021.

[34] 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, Apr. 1991.

[35] S. Boyd, "Proximal Algorithms," p. 113.

[36] P. E. Gill and D. P. Robinson, "A primal-dual augmented Lagrangian," *Computational Optimization and Applications*, vol. 51, no. 1, pp. 1–25, Jan. 2012. [Online]. Available: http://link.springer.com/10.1007/s10589-010-9339-1

[37] J. E. Marsden and M. West, "Discrete mechanics and variational integrators," *Acta Numerica*, vol. 10, pp. 357–514, 2001.

[38] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[39] J. Carpentier, F. Valenza, N. Mansard, and others, *Pinocchio: fast forward and inverse dynamics for poly-articulated systems*, 2015. [Online]. Available: https://stack-of-tasks.github.io/pinocchio

[40] J. Carpentier and N. Mansard, "Analytical Derivatives of Rigid Body Dynamics Algorithms," in *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation, June 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p38.pdf