2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

# Leveraging Proximal Optimization for Differentiating Optimal Control Solvers

Oumayma Bounou[1,2] Jean Ponce[2,3] Justin Carpentier[1,2]

*Abstract*— **Over the past few years, differentiable optimization has gained interest within machine learning, control, and robotics communities. It consists in computing the derivatives of the solutions of a given optimization problem which can then be used in learning algorithms. Until now, dedicated approaches have been proposed to compute the derivatives of various optimization problems (LPs, QPs, SOCPs, etc.). However, these approaches assume in general well-conditioned problems, limiting *de facto* their application to general optimal control problems (OCPs) widely used in robotics. In this work, we focus on the differentiation of solutions to such problems. We notably introduce a differentiable proximal formulation of equality-constrained LQR problems that accurately solves rank-deficient problems. This allows us to compute accurate gradients even in the case of problems that do not satisfy the standard linear independence constraint qualification (LICQ). Because any optimal control problem can be cast as an equality-constrained LQR problem in the vicinity of the optimal solution, we show that our robust LQR derivative computation can be exploited to obtain the derivatives of general optimal control problems. We demonstrate the effectiveness of our approach in dynamics learning and parameter identification experiments in both linear and nonlinear optimal control problems.**

## I. Introduction

Trajectory optimization is a key part of programming robot motions. The task is encoded as a cost term, and physical constraints are encoded as path constraints. Over the past decades, differential dynamic programming (DDP) [1] and the iterative linear quadratic regulator (iLQR) [2] have become widespread and tractable approaches for solving complex robotic problems, ranging from UAVs navigation [3] to legged locomotion [4]. While DDP was originally designed for unconstrained problems, variants have been proposed to account for diverse types of path constraints, ranging from simple bounds on the control inputs [5] to equality [6], [7] and inequality constraints [8], [9], [10].

With recent progress in the development of advanced computational frameworks in machine learning (e.g., PyTorch [11], TensorFlow [12], JAX [13]), differentiable optimization has emerged as a generic approach for computing the derivatives of computational layers involved in mathematical programming problems. It can be used, for instance, to find the optimal design parameters of a robotic mechanism (aka co-design) given a target task [14] or to estimate the sensitivity of an optimal solution to the parameters of the problem [15]. Differentiable optimization generally relies on differentiating the optimality conditions (e.g., the Karush–Kuhn–Tucker conditions) associated with a given problem. In the field of convex optimization, effective approaches are now available for computing derivatives of solutions to standard quadratic programming (QP) instances [16], LQR problems [17], and more general convex programming problems [18], which are very common in control and robotics. However, when deriving the KKT conditions, these approaches often assume that they are well-conditioned at the optimum (e.g., linearly independent constraints qualification, low condition number of the KKT matrix), which cannot be ensured for a large majority of control problems (e.g., inverse or forward dynamics of legged robots making redundant contacts with their environment or singular configurations of robotic arms). This is even more true in machine learning applications, where optimization stages are used as differentiable layers, and the intermediate problems that are being solved during the learning process are not guaranteed to satisfy the required optimality qualifications since they are never enforced during training.

We propose to overcome these limitations by leveraging the proximal method of multipliers [19] to evaluate the derivatives of solutions to optimal control problems robustly. In particular, we develop a generic solution to the computation of these derivatives in the case of the equality-constrained linear quadratic regulator (LQR), which is, as highlighted in [17], the core block required to compute the derivatives of solutions to general nonlinear optimal control problems. Following the experimental validation procedure of [17], we demonstrate that unlike existing solutions, our proximal approach properly handles ill-conditioned system identification problems. We also show that our framework can be used on top of an external nonlinear optimal control solver, enabling our approach to operate on nonlinear problems.

This paper is organized as follows. We first introduce our proximal formulation of equality-constrained LQR problems (Sec. II). We then present our proximal method for differentiating through these problems (Sec. III), and the extension of our approach to the nonlinear case (Sec. IV). In section V, we evaluate and benchmark the proposed solution against alternative solutions of the state of the art on various system identification problems of the literature.

[1]Inria
[2]Département d'Informatique de l'École normale supérieure, (ENS-PSL, CNRS, Inria)
[3]Courant Institute and Center for Data Science, New York University

## II. Problem Statement and Proximal Resolution of Equality-constrained LQR

### A. Problem statement

We address the equality-constrained LQR problem of finding the optimal sequence of states and controls that solve

$$\min_{X,U} \sum_{t=0}^{T-1} l_t(x_t,u_t) + l_T(x_T), \text{ s.t} \begin{cases} x_{t+1} = f_t(x_t,u_t) \\ x_0 = x_0^* \\ c_t(x_t,u_t) = 0 \end{cases}, \tag{1}$$

where $X = \begin{bmatrix} x_0^T & \dots & x_T^T \end{bmatrix}^T$, $U = \begin{bmatrix} u_0^T & \dots & u_{T-1}^T \end{bmatrix}^T$, $l_t$ is a quadratic cost function, $f_t$ is a linear dynamics function and $c_t$ are path constraints. We define the running cost:

$$l_t(x_t,u_t) = \frac{1}{2}(x_t^T Q_t x_t + u_t^T R_t u_t) + q_t^T x_t + w_t^T u_t, \tag{2}$$

the terminal cost:

$$l_T(x_T) = \frac{1}{2}(x_T^T Q_f x_T) + q_T^T x_T, \tag{3}$$

the dynamics:

$$f_t(x_t,u_t) = A_t x_t + B_t u_t + d_t, \tag{4}$$

and the path constraints

$$C_t x_t + D_t u_t - e_t = 0 \text{ and } x_0 = x_0^*. \tag{5}$$

In the case of LQR problems, several approaches [20] can be used to solve problem (1). We follow here the Bellman principle of optimality [21]. Starting from $t = T$, we introduce the value function $V_t$ computed recursively backward in time as:

$$\begin{aligned} V_t(x_t) = \min_{u_t, x_{t+1}} \; & l_t(x_t,u_t) + V_{t+1}(x_{t+1}) \\ \text{s.t. } & x_{t+1} = f_t(x_t,u_t) \text{ and } c_t(x_t,u_t) = 0, \end{aligned} \tag{6}$$

with the terminal condition $V_T(x_T) = l_T(x_T)$. Here, $Q_t$ and $R_t$ are symmetric positive semi-definite matrices, $A_t$, $B_t$, $C_t$ and $D_t$ are general matrices, and $e_t$ and $d_t$ are vectors, for all $t$. $x_0^*$ is the initial state. It is worth mentioning that $C_t$ and $D_t$ might be rank deficient.

### B. Multiple-shooting

Following [22], we also relax the dynamics constraints under Eq. (6) by using multiple shooting [23], which consists in adding auxiliary state variables in order to stabilize the optimization procedure, especially for numerically or intrinsically unstable systems. We introduce an auxiliary variable $y_t$, such that Eq. (6) maps to:

$$\begin{aligned} V_t(x_t) = \min_{u_t, y_t} \; & l_t(x_t,u_t) + V_{t+1}(y_t) \\ \text{s.t. } & y_t = f_t(x_t,u_t) \text{ and } c_t(x_t,u_t) = 0. \end{aligned} \tag{7}$$

The new problem converges to the solution of problem (6) as demonstrated by Schmidt et al. in [24]. In the sequel, we use the usual shorthands $V_x$ and $V_{xx}$ to denote respectively the first and second derivatives of $V_t$, and $V'$ to denote $V_{t+1}$.

At time $t$, we solve problem (7) by solving the equivalent min max problem:

$$V_t(x_t) = \min_{y_t,u_t} \max_{\lambda_t,\nu_t} \; \mathcal{L}_t(x_t,u_t,y_t,\lambda_t,\nu_t), \tag{8}$$

where $\mathcal{L}_t$ is the Lagrangian of the problem:

$$\begin{aligned} \mathcal{L}_t(x_t,u_t,y_t,\lambda_t,\nu_t) = \; & l_t(x_t,u_t) + V'(y_t) \\ & + \lambda_t^T(y_t - f_t(x_t,u_t)) \\ & + \nu_t^T c_t(x_t,u_t). \end{aligned} \tag{9}$$

Here, $y_t$ and $u_t$ are the primal variables and $\lambda_t$ and $\nu_t$ are the dual ones. In the LQR setting, $V_T(x_T) = \frac{1}{2}x_T^T Q_f x_T + q_T^T x_T$, hence $V_t$ is also quadratic for each $t$ [21]. The LQR problem can then be solved using the dynamic programming principle by solving at each time $t$ the linear system obtained by deriving the KKT conditions at optimality:

$$\begin{bmatrix} R_t & 0 & B_t^T & D_t^T \\ 0 & V'_{xx} & -I & 0 \\ B_t & -I & 0 & 0 \\ D_t & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_t \\ y_t \\ \lambda_t \\ \nu_t \end{bmatrix} = - \begin{bmatrix} w_t \\ V'_x \\ A_t x_t + d_t \\ C_t x_t - e_t \end{bmatrix}, \tag{10}$$

where the leftmost matrix is the so-called KKT matrix. Because of the multiple shooting approach, this system has an additional variable $y_t$. However, the KKT matrix has a block-sparse structure, which can be efficiently exploited when solving the system. It should be noted that such a matrix can have very poor conditioning, especially when $V'_{xx}$ or $R_t$ is not positive definite or when $D_t$ is rank-deficient. As mentioned in the introduction, such a scenario is likely to happen in practice, especially in a machine learning context where the cost matrices $R_t$ and $Q_t$ and the path constraints matrix $D_t$ are learnable parameters. To cope with these limitations, we propose solving a proximal reformulation of this problem, which we detail in the following.

### C. Proximal reformulation

As shown by Rockafellar in [19], solving the min-max problem (8) is equivalent to solving the equivalent problem

$$\min_{y_t,u_t} \max_{\lambda_t,\nu_t} \; \mathcal{L}_{t,\rho\mu}(x_t,u_t,y_t,\lambda_t,\nu_t), \tag{11}$$

where $\rho$ and $\mu$ are positive constants,

$$\begin{aligned} \mathcal{L}_{t,\rho\mu}(x_t,u_t,y_t,\lambda_t) = \; & l_t(x_t,u_t) + V_{t+1}(y_t) \\ & + \lambda_t^T(y_t - f(x_t,u_t)) + \nu_t^T c_t(x_t,u_t) \\ & + \frac{\rho}{2}\|u_t - u_t^-\|_2^2 + \frac{\rho}{2}\|y_t - y_t^-\|_2^2 \\ & - \frac{\mu}{2}\|\lambda_t - \lambda_t^-\|_2^2 - \frac{\mu}{2}\|\nu_t - \nu_t^-\|_2^2, \end{aligned} \tag{12}$$

and the four $L_2$ regularization terms vanish over the iterations of a backward/forward procedure for solving problem (11) for all $t$. The so-called *proximal regularization* [25] of the primal variables $u_t$, $y_t$ and dual variables $\lambda_t$, $\nu_t$ is used for solving problems that are not both primally and dually strictly convex, as shown in [7] and [26]. It is central to our approach and will be notably used to solve constrained LQR problems and estimate the sensitivity of their solutions to problem parameters robustly.

Let us now present the backward/forward procedure associated with our approach.

**Backward pass**. For all $t$ backward in time, the new KKT conditions at time $t$ are given by:

$$\begin{bmatrix} R_t+\rho I & 0 & B_t^T & D_t^T \\ 0 & V'_{xx}+\rho I & -I & 0 \\ B_t & -I & -\mu I & 0 \\ D_t & 0 & 0 & -\mu I \end{bmatrix} \begin{bmatrix} du_t \\ dy_t \\ d\lambda_t \\ d\nu_t \end{bmatrix} = - \begin{bmatrix} R_t u_t^- + B_t^T \lambda_t^- + D_t^T \nu_t^- + w_t \\ V'_{xx} y_t^- + V'_x - \lambda_t^- \\ A_t x_t + B_t u_t^- + d_t - y_t^- \\ C_t x_t + D_t u_t^- - e_t \end{bmatrix}, \tag{13}$$

where $dv = v - v^-$, for $v$ in $\{u_t, y_t, \lambda_t, \nu_t\}$. When solving Eq. (13) backward in time, $x_t$ is unknown, but $du_t, dy_t, d\lambda_t$ and $d\nu_t$ can be expressed as affine functions of $x_t$:

$$\begin{cases} du_t &= \Gamma_t x_t + \gamma_t, \\ dy_t &= M_t x_t + m_t, \\ d\lambda_t &= \Xi_t x_t + \xi_t, \\ d\nu_t &= \Omega_t x_t + \omega_t, \end{cases}, \tag{14}$$

and the coefficients $\Gamma_t$, $\gamma_t$, $M_t$, $m_t$, $\Xi_t$ and $\xi_t$ can be obtained by solving the linear system:

$$K_t \begin{bmatrix} \Gamma_t & \gamma_t \\ M_t & m_t \\ \Xi_t & \xi_t \\ \Omega_t & \omega_t \end{bmatrix} = - \begin{bmatrix} 0 & R_t u_t^- + B_t^T \lambda_t^- + D_t^T \nu_t^- + w_t \\ 0 & V'_{xx} y_t^- + V'_x - \lambda_t^- \\ A_t & B u_t^- + d_t - y_t^- \\ -C_t & D_t u_t^- - e_t \end{bmatrix}, \tag{15}$$

where $K_t$ is the KKT matrix from Eq. (13). In the backward pass of the algorithm, we compute the coefficients from Eq. (15) and update the value function's first and second derivatives $V_x$ and $V_{xx}$ for each $t$ (which is straightforward since $V_t$ is quadratic).

**Forward pass**. We update $u_t, y_t, \lambda_t$ and $\nu_t$ for all $t$, forward in time.

$$\begin{cases} u_t^+ = u_t^- + du_t, \\ y_t^+ = y_t^- + dy_t, \\ \lambda_t^+ = \lambda_t^- + d\lambda_t, \\ \nu_t^+ = \nu_t^- + d\nu_t. \end{cases}. \tag{16}$$

We thus solve the LQR problem by iteratively repeating this backward/forward procedure until convergence. Our stopping criterion uses the infinity norm of the gradient of $\mathcal{L}_{\rho,\mu}$

$$\left\| \begin{bmatrix} \nabla_{y,u} \mathcal{L}_{\rho\mu}(y,u) \\ \nabla_{\lambda,\nu} \mathcal{L}_{\rho\mu}(\lambda,\nu) \end{bmatrix} \right\|_\infty \leq \epsilon, \tag{17}$$

where $\mathcal{L}_{\rho\mu}$ stacks $\mathcal{L}_{t,\rho\mu}$ for all $t$ and $\epsilon$ is a fixed tolerance parameter.

## III. Proximal differentiation of equality-constrained LQR

Our objective is to compute derivatives of LQR outputs (i.e., the trajectories $\mathbf{x}$ and $\mathbf{u}$) with respect to the problem parameters (i.e., the dynamics and cost matrices $A_t$, $B_t$, $Q_t$, $R_t$, the path constraints $C_t$, $D_t$, $d_t$, $e_t$, and the initial condition $x_0^*$) to be able to plug the LQR solver as a differentiable layer in a trainable model. Following [17], we first reformulate (6) as the minimization of an equivalent QP problem of the form:

$$\min_X \frac{1}{2} X^T H X + q^T X, \quad \text{s.t } \hat{A} X = \hat{b}, \tag{18}$$

with the following augmented matrices:

$$H = \text{diag}(Q_t, ..., R_t, ...),$$

$$q = \begin{bmatrix} q_0^T & ... & q_T^T & w_0^T & ... & w_{T-1}^T \end{bmatrix}^T,$$

$$\hat{A} = \begin{bmatrix} -I & 0 & 0 & ... & 0 & 0 & 0 & ... & 0 \\ A_0 & -I & 0 & ... & 0 & B_0 & 0 & ... & 0 \\ 0 & A_1 & -I & ... & 0 & 0 & B_1 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ ... & ... & ... & A_{T-1} & -I & 0 & ... & 0 & B_{T-1} \\ C_0 & ... & ... & 0 & 0 & D_0 & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... & ... \\ 0 & ... & ... & C_{T-1} & 0 & D_{T-1} & ... & 0 & 0 \\ 0 & ... & ... & 0 & C_T & 0 & ... & 0 & 0 \end{bmatrix},$$

$$X = \begin{bmatrix} x_0^T & ... & x_T^T & u_0^T & ... & u_{T-1}^T \end{bmatrix}^T,$$

and

$$\hat{b} = - \begin{bmatrix} x_0^{*T} & d_1^T & ... & d_T^T & e_0^T & ... & e_T^T \end{bmatrix}^T.$$

The KKT conditions associated with problem 18 at the optimum correspond to:

$$\hat{K} \begin{bmatrix} X^* \\ \Lambda^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}, \text{ where } \hat{K} = \begin{bmatrix} H & \hat{A}^T \\ \hat{A} & 0 \end{bmatrix} \tag{19}$$

and $\Lambda^*$ is a vector of multipliers associated with the constraints $\hat{A}X = b$ and containing the stacked multipliers $\lambda_t$ and $\nu_t$ from Eq. (9). for all $t$. We drop the "$*$" superscript for clarity in the following.

In machine learning problems, we are looking for derivatives of the form $\partial r / \partial p$, where $r$ is a scalar function and $p$ is a parameter of the problem . Here, $p$ lies in $\mathcal{P}_t = \{A_t, B_t, Q_t, q_t, w_t, C_t, D_t, x_0^*, d_t, e_t\}$. To obtain $\partial r / \partial p$ directly, we use the same trick as in [17] and [27], and obtain expressions for the partial derivatives

$$\begin{aligned} \partial r/\partial A_t &= \lambda_{t+1} G_{x,t}^T + x_t G_{\lambda,t+1}^T, \\ \partial r/\partial B_t &= \lambda_{t+1} G_{u,t}^T + G_{\lambda,t+1} u_t^T, \\ \partial r/\partial Q_t &= x_t G_{x,t}^T, & \partial r/\partial R_t &= G_{u,t} u_t^T, \\ \partial r/\partial C_t &= -x_t G_{\nu,t}^T, & \partial r/\partial e_t &= G_{\nu,t}, \\ \partial r/\partial D_t &= -u_t G_{\nu,t}^T, & \partial r/\partial d_t &= G_{\lambda,t}, \\ \partial r/\partial q_t &= G_{x,t}, & \partial r/\partial w_t &= G_{u,t}, \end{aligned} \tag{20}$$

as functions of a vector $G$ that verifies

$$\hat{K} G = Z, \tag{21}$$

where $Z$ is the vector obtained by stacking all the vectors $\partial r / \partial v_t$ with $v_t$ in $\{x_t, u_t, \lambda_t, \nu_t\}$, for all $t$. The quantities in $Z$ are assumed to be known. In fact, they correspond to the derivatives of $r$ with respect to the outputs of the LQR, i.e., the derivatives of the computational block following the LQR block in the considered computational graph, and can be obtained with back-propagation. We can also write $G$ by stacking the vectors $G_{v,t}$ for all $t$, with $v_t$ in $\{x_t, u_t, \lambda_t, \nu_t\}$ where each vector $G_{v,t}$ is the same size as $v_t$. In [17], Amos et al. notice that $G$ verifies the same equation that the vector $\begin{bmatrix} X^T & \Lambda^T \end{bmatrix}^T$ verifies in Eq. (19). Thus, $G$ is the solution to a

new LQR problem, and as such can be obtained efficiently without explicitly inverting the large KKT matrix $\hat{K}$ in Eq. (21). Instead, we solve a similar LQR problem to the one to which $\mathbf{x}$ and $\mathbf{u}$ are solutions. The KKT matrices of this new problem, which we refer to as the LQR derivatives problem, are the same as the original ones. Only the values of the right-hand side term in the large QP formulation are modified, which correspond to the parameters $q_t$, $w_t$, $d_t$, and $e_t$ in the LQR formulation. Their values for the LQR derivatives problem are:

$$\begin{cases} q_t = \partial r/\partial x_t & w_t = \partial r/\partial u_t \\ d_t = \partial r/\partial \lambda_t & e_t = \partial r/\partial \nu_t \end{cases}. \tag{22}$$

In [17], the authors assume the LICQ conditions are satisfied, whether it is to solve the LQR problem or to compute its derivatives, which implies that $K_t$ is always invertible. However, as discussed in the introduction, this limits the type of control problems the method can be applied to since it is not the case for many of them. Thus, unlike [17], we cope with these limitations by leveraging the proximal formulation introduced in section II-C.

We solve the LQR problem of which $G_{x,t}$, $G_{u,t}$, $G_{\lambda,t}$ and $G_{\nu,t}$ are the solutions and multipliers by solving iteratively until convergence the following linear system for all $t$, backward in time starting from $t = T$:

$$K_t \begin{bmatrix} dG_{u,t} \\ dG_{y,t} \\ dG_{\lambda,t} \\ dG_{\nu,t} \end{bmatrix} = - \begin{bmatrix} R_t G^-_{u,t} + B_t^T G^-_{\lambda,t} + D_t^T G^-_{\nu,t} + \partial r/\partial u_t \\ V'_{xx} G^-_{y,t} + V'_x - G^-_{\lambda,t} \\ A_t G_{x,t} + B_t G^-_{u,t} + \partial r/\partial \lambda_t - G^-_{y,t} \\ C_t G_{x,t} + D_t G^-_{u,t} - \partial r/\partial \nu_t \end{bmatrix}, \tag{23}$$

where

$$K_t = \begin{bmatrix} R_t + \rho I & 0 & B_t^T & D_t^T \\ 0 & V'_{xx} + \rho I & -I & 0 \\ B_t & -I & -\mu I & 0 \\ D_t & 0 & 0 & -\mu I \end{bmatrix} \tag{24}$$

and obtain new updates

$$\begin{bmatrix} G_{u,t} \\ G_{y,t} \\ G_{\lambda,t} \\ G_{\nu,t} \end{bmatrix} = \begin{bmatrix} G^-_{u,t} + dG_{u,t} \\ G^-_{y,t} + dG_{y,t} \\ G^-_{\lambda,t} + dG_{\lambda,t} \\ G^-_{\nu,t} + dG_{\nu,t} \end{bmatrix}. \tag{25}$$

This allows us to find the optimal variable $G$ accurately, even in ill-conditioned problems (e.g., rank-deficient matrices $K_t$). This is extremely important in practice since the gradients of interest, $\partial r/\partial p$ where $p$ is in $\mathcal{P}$, are functions of the values of $G$ (equation (20)). Thus, any inaccurate solution $G$ would lead to back-propagating incorrect gradients $\partial r/\partial p$, resulting in unstable training procedures.

## IV. Extension to nonlinear cases

When $f$, $l$, and $c$ are nonlinear, problem (1) corresponds to an equality-constrained nonlinear optimal control problem. A standard method for solving it is the so-called iterative LQR (iLQR) algorithm [2], performing a cascade of LQR problems by linearizing the dynamics and making a quadratic approximation of the cost function around the nominal state and control trajectories. In the case of equality-constrained nonlinear optimal control problems, the approximations can be written as follows:

$$\min_{\delta X, \delta U} \sum_{t=0}^{T-1} \frac{1}{2} \delta x_t^T L_{t,xx} \delta x_t + \frac{1}{2} \delta u_t^T L_{t,uu} \delta u_t + x_t^T L_{t,xu} \delta u_t + l_{t,x}^T \delta x_t + l_{t,u}^T \delta u_t \tag{26}$$

$$\text{s.t. } F_{t,x} \delta x_t + F_{t,u} \delta u_t + f_0 = 0, \tag{27}$$
$$C_{t,x} \delta x_t + C_{t,u} \delta u_t + c_0 = 0,$$

where $\delta x_t$ and $\delta u_t$ stand for the step directions and $\mathcal{D}_{l,f} = \{L_{t,xx}, L_{t,uu}, L_{t,xu}, l_{t,x}, l_{t,u}, F_{t,x}, F_{t,u}, C_{t,x}, C_{t,u}\}$ is the set of first and second-order derivatives of $l$, $f$ and $c$ respectively.

iLQR-based approaches for solving unconstrained and constrained problems of this kind [5], [9], [22] proceed iteratively until a convergence criterion is met. At the optimum, the KKT conditions of the nonlinear constrained optimal control problem (1) correspond to those of the last LQR problem solved in the iLQR. Thus, the derivatives of problem (1) in the general case can be obtained with the method described in III. The solution extends to OCPs containing inequality constraints, since active inequality constraints at the optimum play the role of pure equality constraints.

Finally, if we assume that the dynamics $f$, constraints $c$, and cost function $l$ are parameterized by some parameter $\theta$, then it follows that the set of first and second-order derivatives $\mathcal{D}_{l,f}$, and specifically the one at optimality $\mathcal{D}^*_{l,f}$ will also be parameterized by this parameter $\theta$. In a learning framework, the derivative of interest, i.e., $\partial r/\partial \theta$, with $r$ some scalar loss function over the parameters can thus be obtained by applying the chain rule, leading to:

$$\frac{\partial r}{\partial \theta} = \sum_{p \in \mathcal{D}^*_{l,f}} \frac{\partial r}{\partial p} \frac{\partial p}{\partial \theta}, \tag{28}$$

where the derivatives $\partial r/\partial p$ can be obtained with the method described in paragraph III, and the derivatives $\partial p/\partial \theta$ are obtained with backpropagation.

Overall, our approach is quite general and we can use any off-the-shelf iLQR solver to get the set of derivatives of the OCP at optimality $\mathcal{D}^*_{l,f}$ to formulate the quadratic cost problem with linear equality constraints at optimality, which we then solve with our approach.

## V. Experiments

We first evaluate our method for solving LQR problems with terminal constraints. We then evaluate it on cost and parameter identification problems for both linear (LQR) and nonlinear systems (pendulum and cartpole). The code associated with this paper, written in Python, will be released as open-source.

### A. LQR problems with terminal constraints

We solve an LQR problem with a terminal equality constraint (Sec. II-A) following the approach described in section II-C. We compare the solutions of both our solver and CVXPY [28], [29], which is based on OSQP [30]. We run both solvers on three sets of parameter sizes $\{n, d, T\}$, where

TABLE I

SOLVERS PERFORMANCE: COMPARISON OF LQR PROBLEMS WITH TERMINAL EQUALITY CONSTRAINTS.

| Parameters | solver | success | feasibility | distance to goal |
|---|---|---|---|---|
| $T = 20$, $n = 15$, $d = 3$ | cvxpy | 2 | $6.10^{-6}$ | $6.10^{-6}$ |
| | ours | **93** | $\mathbf{9.10^{-9}}$ | $\mathbf{9.10^{-9}}$ |
| $T = 20$, $n = 10$, $d = 2$ | cvxpy | 32 | $2.10^{-8}$ | $2.10^{-8}$ |
| | ours | **98** | $\mathbf{4.10^{-9}}$ | $\mathbf{4.10^{-9}}$ |
| $T = 20$, $n = 15$, $d = 5$ | cvxpy | **100** | $9.10^{-12}$ | $9.10^{-12}$ |
| | ours | **100** | $\mathbf{2.10^{-14}}$ | $\mathbf{3.10^{-15}}$ |

$n$ is the system dimension, $d$ is the control dimension, and $T$ is the time horizon. For each set of parameter sizes, we run 100 experiments with randomly generated time-invariant LQR problems. The dynamics matrix $A$ is forced to have all singular values lower than 1, and $B$ is a matrix with random coefficients sampled uniformly in $[0, 1]$. The cost matrices $Q$ and $R$ are set to respectively $10^{-2}I_n$ and $10^{-1}I_d$. All experiments are run on a single CPU. A solver is considered successful when both primal and dual constraints are satisfied. Feasibility denotes the infinity norm on primal constraints (dynamics constraints $\|x^*_{t+1} - Ax^*_t - Bu^*_t\|_\infty$). Distance to goal is the average infinite norm $\|x_T - x^*_T\|_\infty$, with $x^*_T$ the target terminal constraint.

Tab. I reports the average results of our experiments. In the first and second rows, our solver converges to a solution for almost every experiment, while CVXPY only converges to a solution for up to 32% of them on average. We explain this difference by the fact that for these problems, the KKT matrix involved in the last step of the dynamic programming problem has a condition number of $10^{17}$ in the unregularized formulation, and of $10^7$ in our regularized approach. In examples where both solvers converge, the proposed approach converges to a more accurate solution. This first set of results shows both increased robustness and better accuracy of the proximal solver against a more classic approach for solving LQR problems.

We also report the average results on well-conditioned cases (third row). In such cases, both solvers converge to equally good solutions, but CVXPY is much faster since our implementation is in an interpreted language (Python) that could easily be moved to a compiled one (e.g., C++) for much better efficiency.

### *B. System identification: LQR problems*

We reproduce the system identification experimental setting of [17]. Given optimal trajectories in states and controls of systems with linear dynamics and quadratic cost, the goal is to identify these dynamics and cost parameters. Formally, we solve the LQR problem of equations (2)-(5) with $q_t = 0$ and $w_t = 0$:

$$\begin{aligned} \min_{X,U} \ & \frac{1}{2}\sum_{t=0}^{T-1}(x_t^T Q x_t + u_t^T R u_t) + \frac{1}{2}x_T^T Q_f x_T, \\ \text{s.t } & x_{t+1} = Ax_t + Bu_t \text{ for } t \text{ in } 0, \dots, N-1, \\ & x_0 = x^*_0. \end{aligned} \tag{29}$$

TABLE II

SUCCESS RATE OF IDENTIFICATION METHODS: WE COMPARE LEVENBERG-MARQUARDT AND RMSPROP OPTIMIZATION SUCCESS RATES (IN %).

| | T=5, n=3, d=3 | T=20, n=3, d=3 | T=10, n=5, d=2 |
|---|---|---|---|
| RMSProp | 21 | 20 | 21 |
| LM | **91** | **85** | **90** |

We observe $M$ trajectories $[x_0^{*,i}, \dots x_T^{*,i}]$ and $[u_0^{*,i}, \dots, u_{T-1}^{*,i}]$ ($i$ in $1, \dots, M$) that are optimal solutions to problem (29) with different initial conditions $x_0^*$ (chosen at random). The states $x_t$ are vectors of size $n$ (between 2 and 10), the controls $u_t$ are vectors of size $d$ (between 3 and 10), and the control horizon is $T$ (between 5 and 20). Our goal is to identify the dynamics matrices $(A, B)$ and the cost matrices $(Q, Q_f, R)$. In other words, we want to estimate $\theta$, where $\theta$ can be any of $A, B, Q, Q_f, R$, or a combination of two or more of these matrices. We solve:

$$\min_\theta \sum_{i=1}^{M}\sum_{t=0}^{T} \|x_t^{*,i} - x_t^i(\theta)\|_2^2 + \|u_t^{*,i} - u_t^i(\theta)\|_2^2 \tag{30}$$

where $x_t^i$ and $u_t^i$ are solutions of the LQR problem parameterized by $\theta$ with initial condition $x_0^{*,i}$. Using the same notations as in [17], we define the optimal trajectory vector $i$, $\tau^{*,i}$ as

$$\tau^{*,i} = \begin{bmatrix} x_0^{*,i^T} & \dots & x_T^{*,i^T} & u_0^{*,i^T} & \dots & u_{T-1}^{*,i^T} \end{bmatrix}^T. \tag{31}$$

Here, $\tau^{*,i}$ is a vector of size $p = (T+1)n + Td$. The problem is now reduced to solving

$$\min_\theta \sum_{k=0}^{Mp} \|r_k(\theta)\|_2^2, \tag{32}$$

where $r_k(\theta) = \Gamma^*_k - \Gamma_k$, $\tau^{*,i}_k$ (respectively $\tau^i_k(\theta)$) is the $k$-th component of vector $\tau^{*,i}$(respectively $\tau^i$), and $\Gamma^*_k$(respectively $\Gamma_k$) contains stacked vectors $\tau^{*,i}$ (respectively $\tau^i$). Problem (32) is a non-linear least-squares problem that can be solved using methods such as Gauss-Newton (GN) [31] or Levenberg-Marquardt (LM) [32]. When a step in a solution of an optimal control problem boils down to least squares, stochastic gradient descent should not be used for this step (as expected from the optimization literature), which partly explains what happens in [17], where the identification experiments fail in half the trials when using gradient descent to optimize Eq. (30).

Table II shows results averaged on 100 experiments. An experiment is considered successful if the identification error $\|\theta - \theta^*\|_\infty$ reaches the threshold $5.10^{-6}$ in less than 50 iterations of the LM algorithm or 2000 epochs of RMSProp [33]. We see that the identification succeeds 85% of the time when using a least-squares method, while it only succeeds 20% of the time when using RMSProp, which demonstrates the ineffectiveness of stochastic gradient methods in this setting. Note that for the experiments using RMSProp, we have selected the hyperparameters (learning rate and batch size), which achieved the lowest errors. It

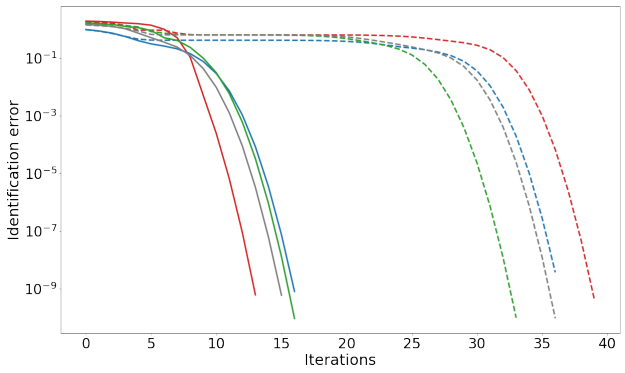


Fig. 1. **Identification error** on identifying the matrices $A$ and $B$ with $Q = 10^{-4}I_n$. Pairs of curves with the same colors are identification experiments on the same problem parameters solved using different solvers: diff-mpc in dashed lines and ours in solid lines. The same convention is used in all figures.

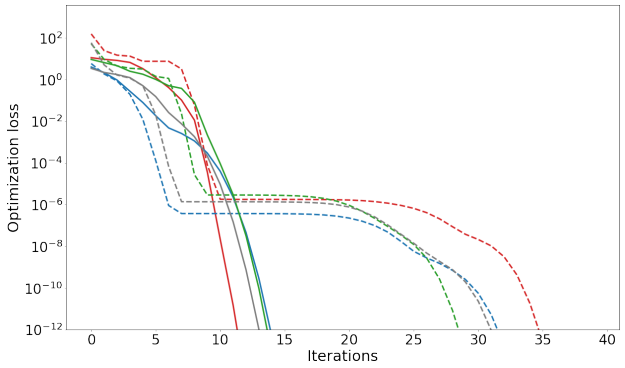


Fig. 2. **Optimization loss** on identifying the matrices $A$ and $B$ with $Q = 10^{-4}I_n$.

should be noted, however, that implementing Levenberg-Marquardt-like methods requires computing the full Jacobian matrix of the residual function $r$ (instead of just Jacobian-vector products as in SGD methods), which is a function that scales linearly with $T$, $n$ and $d$.

In the following experiments, the identification problem (32) is solved using the LM method for both our solver and the solver from [17]. When using our solver, the proximal parameters $\rho$ and $\mu$ from equation (12) are set to $10^{-8}$. Problem (11) still converges to the same solution with higher values of these parameters, provided more iterations are run. Strategies where the parameters $\rho$ and $\mu$ are updated over the iterations like the bound-constrained lagrangian (BCL) method from [34] may be adopted to enhance convergence speed when it is a limitation, but it has not been the case in our experiments.

Figures 1 and 2 show the system identification error and optimization loss as functions of the number of iterations using both the solver from [17], which we refer to as diff-mpc, and our solver on 4 trials (100 experiments were run, each with a different set of parameters to identify, but we only show 4 randomly selected ones here). In this experiment, the parameters to identify are the dynamics and control matrices $A$ and $B$. The control cost matrix $R$ is set to the identity

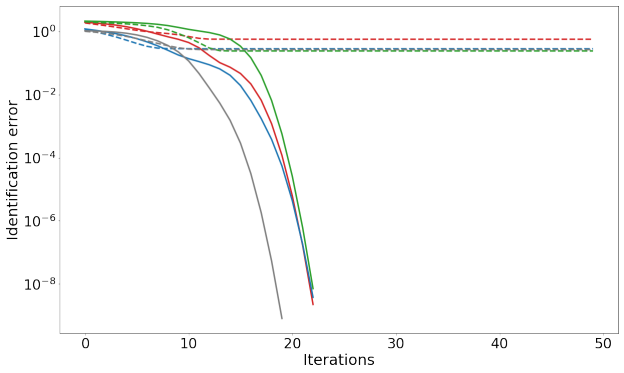


Fig. 3. **Identification error.** Identification of $Q$.

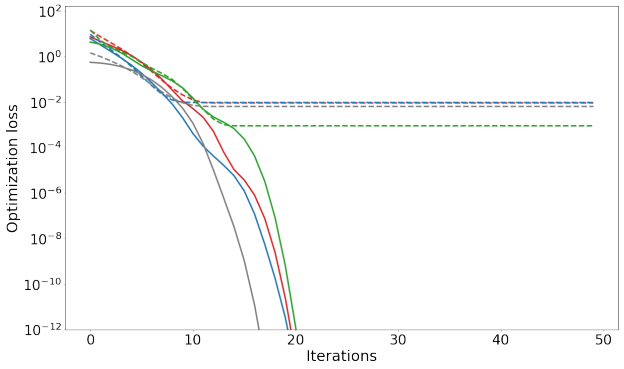


Fig. 4. **Optimization loss.** Identification of $Q$.

matrix, and the state cost matrix $Q$ is set to $10^{-4}I_n$. In this experiment, both solvers converge, but our regularized solver converges in half as many iterations on all the trials.

Figures 3 and 4 show the system identification error and optimization loss as functions of the number of iterations using both solvers again, on experiments where the parameter to identify is the matrix $Q$ (the identity matrix). The approach of [17] fails on all trials, demonstrating the importance of regularizing the LQR solver to avoid performing optimization steps with wrong gradients.

### C. System identification: nonlinear dynamics

We demonstrate the effectiveness of our approach in identifying parameters of a nonlinear system from observed state and control trajectories. We consider a pendulum parameterized by its mass and its pole length and a cartpole parameterized by its pole mass, its cart mass, and its pole length. We seek to recover these parameters for both systems. The state of the pendulum at time $t$ is defined by $s_t = [\cos\theta_t, \sin\theta_t, \dot{\theta}_t]$, and the state of the cartpole by $s_t = [x_t, \dot{x}_t, \cos\theta_t, \sin\theta_t, \dot{\theta}_t]$ (translation over the $x$ axis). We have conducted experiments with pendulums and cartpoles with 10 various sets of parameters and report the losses for 4 of them for each system (see table III). The state cost matrix is set to $Q = 10^{-3}I_n$ for the pendulum and to $Q = 10^{-1}I_n$ for the cartpole. The control cost matrix is set to $R = 10^{-1}I_d$ for both systems. A linear cost term is added to bring the pole upward (for both systems) and the cart centered at the

origin (for the cartpole).

Here, the problem is also a nonlinear least-squares one, and it can be formulated as in Eq. (32), with $\theta = \{m, l\}$ for the pendulum, and $\theta = \{m, m_c, l\}$ for the cartpole. We solve it using the LM method for both solvers and report the results in Figures 5, 6, 7 and 8. Here again, the proximal parameters $\rho$ and $\mu$ are set to $10^{-8}$ when using our solver, similar to the experiments in Sec. V-A. We see that, unlike diff-mpc, our method converges to the ground truth parameters in all experiments. When both methods converge, ours converges in fewer iterations to a better solution.

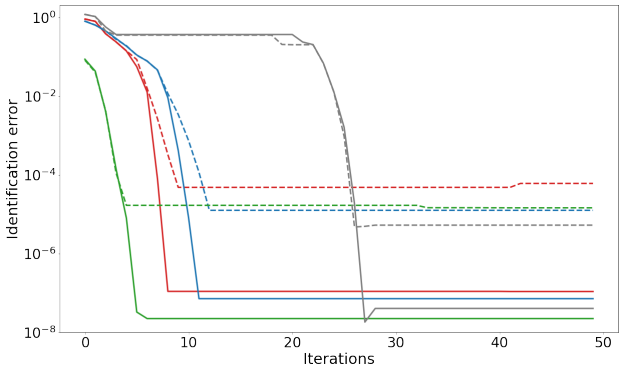


Fig. 5. **Pendulum. Identification error.** Identification of $m$ and $l$.

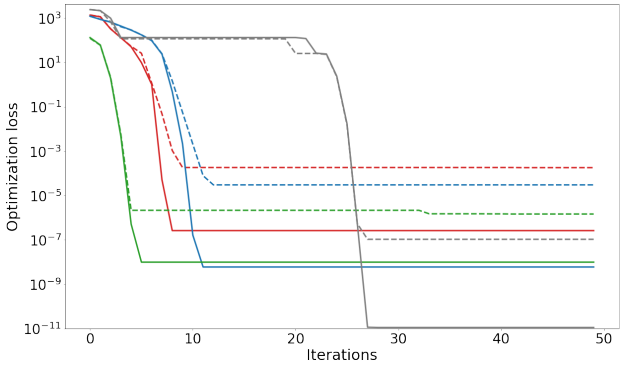


Fig. 6. **Pendulum. Optimization loss.** Identification of $m$ and $l$.

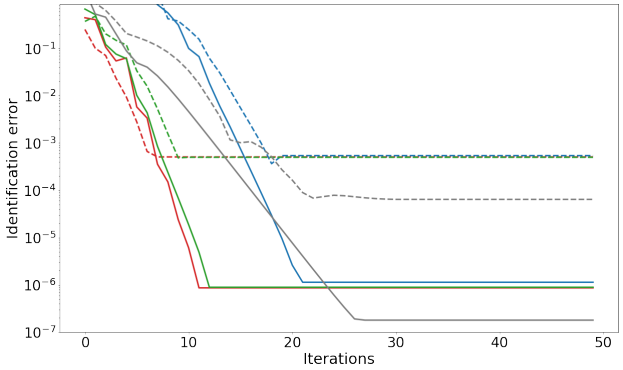


Fig. 7. **Cartpole. Identification error.** Identification of $m$, $m_c$ and $l$.

TABLE III

GROUND TRUTH PARAMETERS $(m, l)$ AND $(m_c, m, l)$ FOR THE IDENTIFICATION EXPERIMENTS RESPECTIVELY ON THE PENDULUM AND CARTPOLE. MASSES ARE IN $kg$ AND LENGTHS ARE IN $m$. THE COLORS CORRESPOND TO THE ONES IN THE ERROR AND LOSS FIGURES.

| | Red – | Green – | Blue – | Grey – |
|---|---|---|---|---|
| Pendulum | $(1, 1)$ | $(2, 0.5)$ | $(1.5, 1)$ | $(0.5, 1)$ |
| Cartpole | $(1, 0.1, 0.5)$ | $(1, 0.2, 0.5)$ | $(1, 0.5, 0.5)$ | $(2, 1, 1)$ |

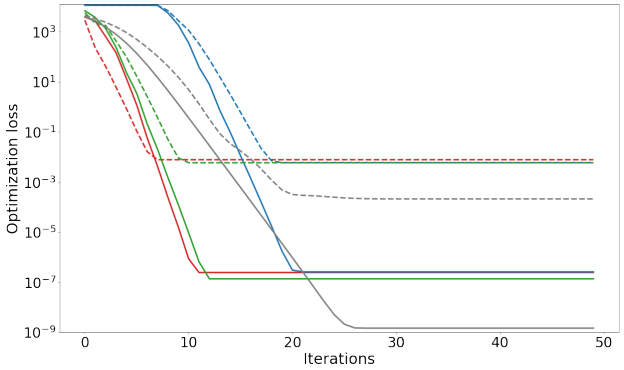


Fig. 8. **Cartpole. Optimization loss.** Identification of $m$, $m_c$ and $l$.

## VI. CONCLUSION

We have introduced a regularized differentiable equality-constrained LQR solver with a generic formulation that handles path constraints. When used in learning frameworks, our solver is robust to ill-conditioned problems and performs better on system identification experiments than the unregularized approach introduced in [17]. Since our formulation of the LQR problem is generic, it can also be applied to differentiate accurately through more general optimal control problems. Future work should focus on experiments on real robotic systems to demonstrate the effectiveness and robustness of this approach on real-world robotic tasks.

## VII. ACKNOWLEDGEMENTS

We warmly thank Wilson Jallet, Quentin Le Lidec, Etienne Arlaud, Eloise Berthier, and Antoine Bambade for fruitful discussions. This work was supported in part by the HPC resources from GENCI-IDRIS (Grand 2023-AD011011263R3), the European Union through the AGIMUS project (GA no.101070165), the Inria/NYU collaboration, the Louis Vuitton/ENS chair on artificial intelligence, the French government under the management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program (reference ANR19-P3IA0001, PRAIRIE 3IA Institute) and the ANR JCJC project NIMBLE (ANR-22-CE33-0008).

## REFERENCES

[1] D. Q. Mayne, "Differential Dynamic Programming–A Unified Approach to the Optimization of Dynamic Systems," vol. 10 of *Control and Dynamic Systems*, pp. 179–254, Academic Press, 1973.

[2] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *ICINCO (1)*, pp. 222–229, Citeseer, 2004.

[3] J. Marti-Saumell, J. Solà, C. Mastalli, and A. Santamaria-Navarro, "Squash-box feasibility driven differential dynamic programming," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7637–7644, IEEE, 2020.

[4] 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)*, pp. 1–9, IEEE, 2018.

[5] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175, IEEE, 2014.

[6] M. Giftthaler and J. Buchli, "A projection approach to equality constrained iterative linear quadratic optimal control," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 61–66, IEEE, 2017.

[7] S. El Kazdadi, J. Carpentier, and J. Ponce, "Equality constrained differential dynamic programming," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8053–8059, IEEE, 2021.

[8] 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.

[9] 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)*, pp. 7674–7679, IEEE, 2019.

[10] W. Jallet, A. Bambade, N. Mansard, and J. Carpentier, "Constrained Differential Dynamic Programming: A primal-dual augmented Lagrangian approach," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Kyoto, Japan), Oct. 2022.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[13] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018.

[14] G. Fadini, T. Flayols, A. Del Prete, N. Mansard, and P. Souères, "Computational design of energy-efficient legged robots: Optimizing for size and actuators," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9898–9904, IEEE, 2021.

[15] A. Oshin, M. Houghton, M. Acheson, I. Gregory, and E. Theodorou, "Parameterized Differential Dynamic Programming," in *Proceedings of Robotics: Science and Systems*, (New York City, NY, USA), June 2022.

[16] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, pp. 136–145, PMLR, 2017.

[17] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable mpc for end-to-end planning and control," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[18] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[19] 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.

[20] S. Boyd, "Lqr." "https://stanford.edu/class/ee363/lectures.html", 2008.

[21] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.

[22] W. Jallet, N. Mansard, and J. Carpentier, "Implicit Differential Dynamic Programming," in *International Conference on Robotics and Automation (ICRA 2022)*, (Philadelphia, United States), IEEE Robotics and Automation Society, May 2022.

[23] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[24] M. Schmidt, N. Roux, and F. Bach, "Convergence rates of inexact proximal-gradient methods for convex optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[25] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, p. 127–239, Jan. 2014.

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

[27] Q. Le Lidec, I. Kalevatykh, I. Laptev, C. Schmid, and J. Carpentier, "Differentiable simulation for physical system identification," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3413–3420, 2021.

[28] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[29] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[30] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: an operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[31] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization*. Springer Science & Business Media, 2008.

[32] J. J. Moré and D. C. Sorensen, "Computing a trust region step," *Siam Journal on Scientific and Statistical Computing*, vol. 4, pp. 553–572, 1983.

[33] G. Hinton, N. Srivastava, and K. Swersky, "Rmsprop." "https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf", 2016.

[34] A. R. Conn, N. I. M. 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, no. 2, pp. 545–572, 1991.