Accelerating Hybrid Systems Differential Dynamic Programming

John N. Nganga[*] **and Patrick M. Wensing**
Department of Mechanical Engineering
University of Notre Dame
Notre Dame, IN 46556
Email: {jnganga,pwensing}@nd.edu

This letter presents approaches that reduce the computational demand of including second-order dynamics sensitivity information into optimization algorithms for robots in contact with the environment. A full second-order Differential Dynamic Programming (DDP) algorithm is presented where all the necessary dynamics partial derivatives are computed with the same complexity as DDP's first-order counterpart, the iterative Linear Quadratic Regulator (iLQR). Compared to linearized models used in iLQR, DDP more accurately represents the dynamics locally, but it is not often used since the second-order partials of the dynamics are tensorial and expensive to compute. This work illustrates how to avoid the need for computing the derivative tensor by instead leveraging reverse-mode accumulation of derivatives, extending previous work for unconstrained systems. We exploit the structure of the contact-constrained dynamics in this process. The performance of the proposed approaches is benchmarked with a simulated model of the MIT Mini Cheetah executing a bounding gait.

1 Introduction

In comparison to their biological counterparts, legged robots are yet to be as mobile and dexterous. Animals devise and execute fast and fluid movements on the fly despite having numerous degrees of freedom (DoF). This capability is most beneficial in new, unknown, and uneven terrain – where the fast and fluid movements are tailored to the novel context. The scientific gap for legged robots to achieve this goal is, in part, due to challenges from the robot's highly nonlinear dynamics and the curse of dimensionality from many DoFs. Toward endowing robotic platforms with the ability to traverse over ground like their biological counterparts, current approaches often cast the robotic locomotion problem as a trajectory optimization (TO) problem and solve for the optimal control tape. The resulting control inputs ideally allow the robot to achieve some desired motion while maintaining energetic efficiency and respecting several constraints.

However, the high-dimensional state space and highly coupled nonlinear dynamics of robots often render the solution of a whole-body optimal control problem (OCP) impractical. Conventionally, the solutions to these OCPs rely on model reduction to limit the dimensionality and nonlinearity of the problem. For example, [1, 2] use simple dynamical models (i.e., template models [3]) that encode the salient features of the full dynamical model. The authors [1, 2, 4] use the Spring-loaded inverted pendulum (SLIP) template model to optimize the gait of a bipedal robot, an articulated robotic leg, and a humanoid, respectively. The work of [5] uses the evolution of the angular momentum and the center of mass position, typically known as centroidal dynamics, for trajectory generation for a humanoid. The work of [6] uses a single-rigid-body approximation of the full model that ignores the leg dynamics while reducing the OCP to a convex problem. Finally, [7] uses the zero-moment point (ZMP) to generate reference trajectories for a quadruped. While template models significantly simplify the optimization problem, they do not reason about all pertinent features of the robotic platform. The effect is that the resulting motions cannot reach the performance limits of the full system as compared to motions generated via whole-body motion planning.

Whole-body motion planning considers the full dynamics model of the robot and solves a finite-horizon TO problem. The computational burden to solve this problem is significantly greater than when adopting a template model. With the advent of more computational power, however, the transition to whole-body TO is steadily gaining popularity. For example, the authors in [8] used a whole-body trajectory optimizer based on Differential Dynamic Programming (DDP) to control a 22-DoF humanoid robot. Rather than optimizing all control variables at once, DDP exploits the temporal sparsity of the OCP and solves a sequence of smaller optimization problems at each point along the horizon. Further, DDP enforces trajectory continuation between time points, making it a shooting method. The effect is that trajectories are dynamically valid at any iteration of the DDP optimization process, which encourages their usability online before the optimizer converges. In addition to a

[*]Corresponding Author: jnganga@nd.edu

feed-forward control tape, DDP also outputs a locally optimal feedback policy, which can be used to handle disturbances [9]. As originally described [10], DDP does not handle constraints. However, several recent modifications to DDP have been proposed that address control limits [11] and general state/control constraints [12, 13]. Further, the works of [12, 14–16] have extended the algorithm from smooth dynamical systems to hybrid systems that have sequences of smooth modes with reset maps determining transitions between them. In this work, we consider the extension of DDP to systems undergoing rigid contacts with the environment in the same fashion as [14], but with the mode timings fixed. Beyond addressing hybrid dynamics that arise due to different contact configurations, these methods can also be applied to adopt coarsening modeling abstractions across the planning horizon [17], which can reduce the computation demand of whole-body planning.

To perform TO, DDP considers second-order approximations of the dynamics model. However, in practice (e.g., [8, 15, 18]), many researchers have opted to use a first-order dynamics approximation due to its faster evaluation time, giving rise to the iterative Linear Quadratic Regulator (iLQR). While the second-order dynamics information retains higher fidelity to the full model locally, it is represented by a rank three tensor and is expensive to compute. Our previous work [19] avoided the vector-tensor operation in DDP by presenting an algorithm to calculate the necessary second-order partials efficiently. Reverse-mode accumulation of derivatives was leveraged to achieve that effect for smooth dynamical systems. In this work, we extend [19] to constrained rigid-body systems with a focus on systems in rigid contact with the environment. Despite this application focus, much of our development readily applies to other constraints.

1.1 Specific Contributions

This letter presents several advances that accelerate DDP for use with systems in rigid contact with the environment, as in legged robots. We (I) expose the structure of the second-order dynamics partials needed in DDP, paving the way toward an efficient computation strategy that avoids calculating the full second-order dynamics derivative tensor. The structural form uncovered motivates us to (II) *extend* the Recursive-Newton-Euler Algorithm (RNEA) to support our computation of contact-constrained dynamics partials. The resulting algorithm is called the modified RNEA for contacts (mRNEAc). Finally, we (III) use reverse-mode Automatic Differentiation (AD) tools with mRNEAc to accelerate the computation of second-order partials needed in DDP as compared to conventional approaches. The final outcome is that the proposed methods allow for the computation of the needed second-order dynamics information in the same computational complexity as in iLQR.

The remainder of this letter is organized as follows. Section 2 provides rigid-body dynamics preliminaries and discusses a DDP algorithm for hybrid systems with fixed mode sequences. Sections 3 and 4 detail the proposed approach for reducing the computational demand of including second-order dynamics information in DDP with contact-constrained systems. Section 5 presents results and an application to optimizing a quadruped bounding gait. Finally, a discussion is provided in Section 6, with Section 7 concluding and offering future directions.

2 Background: Trajectory Optimization

This work considers the efficient solution of a finite-horizon OCP for a hybrid system model with a fixed mode sequence. Section 2.1 reviews the formulation for unconstrained and constrained rigid-body dynamics. Finally, in Section 2.2, we introduce our DDP formulation for hybrid dynamics.

2.1 Dynamics Preliminaries

The inverse dynamics (ID) of a fully-actuated and unconstrained rigid-body system can be computed with the RNEA [20, 21] as:

$$\begin{aligned}\tau &= \mathrm{RNEA}(\mathrm{model}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g) \\ &= \mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}),\end{aligned}$$

where $\mathbf{H}$, $\mathbf{q}, \tau$, and $\mathbf{a}_g$ denote the inertia matrix, generalized coordinates, joint torques, and gravitational vector, respectively. The term $\mathbf{h}$ is the joint-space bias vector, and it groups the Coriolis and centrifugal vector $\mathbf{C}\dot{\mathbf{q}}$ and gravitational components τ_g together. The RNEA can be evaluated with $\mathcal{O}(n)$ complexity, where n is the number of DoFs in the model. The RNEA is a two-pass algorithm where the forward pass propagates the velocity and acceleration terms, and the backward pass determines forces at the joints.

Consider a model experiencing contacts with the environment, which impose constraints on the free dynamics. Given the contact status (also called the contact mode herein), the dynamics can be determined via:

$$\underbrace{\begin{bmatrix} \mathbf{H} & -\mathbf{J}_c^\top \\ -\mathbf{J}_c & 0 \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix}}_{\nu} = \underbrace{\begin{bmatrix} \mathbf{S}^\top \tau - \mathbf{h} \\ \dot{\mathbf{J}}_c \dot{\mathbf{q}} \end{bmatrix}}_{\Psi}, \tag{1}$$

where $\mathbf{S}$ denotes the selector matrix that picks out the actuated joints. The terms $\mathbf{J}_c \in \mathbb{R}^{n_c \times n}$ and $\lambda \in \mathbb{R}^{n_c}$ represent the contact Jacobian and the corresponding contact forces, where $n_c = 3$ for a point contact. The matrix $\mathbf{K}$ on the left is typically known as the KKT matrix [22]. We group the joint acceleration and contact forces as $\nu = [\ddot{\mathbf{q}}^\top \;\; \lambda^\top]^\top$ and group the right-hand side of (1) as a vector Ψ such that the solution for (1) can be given as

$$\begin{bmatrix} \mathfrak{F}(\mathbf{q}, \dot{\mathbf{q}}, \tau) \\ \mathfrak{g}(\mathbf{q}, \dot{\mathbf{q}}, \tau) \end{bmatrix} \triangleq \nu = \mathbf{K}^{-1}\Psi, \tag{2}$$

where $\mathfrak{F}$ represents the realized continuous joint acceleration function (i.e., forward dynamics) and $\mathfrak{g}$ is the realized contact forces function. In flight (i.e., no contacts, $n_c = 0$), (2) resolves to the unconstrained dynamics $\mathfrak{F} \triangleq \ddot{\mathbf{q}} = \mathbf{H}^{-1}(\mathbf{S}^\top \tau - \mathbf{h})$, with $\mathfrak{g} = \emptyset$.

For impact events, the dynamics take the same form as (1) with the exception that velocities change instantaneously at each impact event such that

$$\begin{bmatrix} \mathbf{H} & -\mathbf{J}_c^\top \\ -\mathbf{J}_c & 0 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}^+ \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{H}\dot{\mathbf{q}}^- \\ e\mathbf{J}_c\dot{\mathbf{q}}^- \end{bmatrix} \tag{3}$$

where $\dot{\mathbf{q}}^+$ and $\dot{\mathbf{q}}^-$ denote the pre- and post- impact velocities, respectively. The term, $\hat{\lambda}$, denotes the impulsive force that acts upon contact and e denotes the coefficient of restitution. We presume perfect inelastic collision such that $e = 0$. Since actuators cannot generate impulsive torques, control inputs do not appear in the impact equation.

2.2 Hybrid Systems DDP Algorithm

Consider a model with state $\mathbf{x} = [\mathbf{q}^\top \ \dot{\mathbf{q}}^\top]^\top$, control input $\mathbf{u} = \tau$, and ground reaction forces λ. For each contact mode, the continuous state and control trajectories can be discretized using a numerical integration scheme, with forward Euler assumed here:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \triangleq \mathbf{x}_k + h \begin{bmatrix} \dot{\mathbf{q}} \\ \mathfrak{F}(\mathbf{q}, \dot{\mathbf{q}}, \tau) \end{bmatrix}, \\ \lambda_k &= \mathfrak{g}(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{4}$$

where h is the integration step size.

DDP/iLQR is used to solve an OCP with a fixed sequence of m modes such that the total cost $\mathbf{J}(\mathbf{U})$ is a summation across each mode, given as:

$$\mathbf{J}(\mathbf{U}) = \sum_{i=1}^{m} \left[\Phi_i(\mathbf{x}_{N_{i+1}^-}) + \sum_{k=N_i}^{N_{i+1}-1} \ell_i(\mathbf{x}_k, \mathbf{u}_k, \lambda_k) \right]. \tag{5}$$

Here, ℓ_i is the running cost, $\mathbf{U} = [\mathbf{U}_1, , \ldots, \mathbf{U}_m]$ is the control sequence over the horizon, $\mathbf{U}_i$ is the control sequence in mode i. The term Φ_i is the terminal cost at the end of mode i, which runs from index N_i to N_{i+1}, with N_i^- and N_i^+ denoting the instants before and after the transition to mode i. To obtain an optimal trajectory, (5) is minimized across modes as

$$\min_{\mathbf{U}} \ \mathbf{J}(\mathbf{U}) \tag{6a}$$
$$\text{subject to } \mathbf{x}_{k+1} = \mathbf{f}_i(\mathbf{x}_k, \mathbf{u}_k) \tag{6b}$$
$$\lambda_k = \mathfrak{g}_i(\mathbf{x}_k, \mathbf{u}_k) \tag{6c}$$
$$\mathbf{x}_{\mathbf{N}_i}^+ = \mathbf{f}_{\text{Imp}}(\mathbf{x}_{\mathbf{N}_i}^-) \tag{6d}$$
$$\mathbf{g}_c(\mathbf{x}_{N_i}^-) = 0 \tag{6e}$$
$$\varphi(\mathbf{u}_k, \lambda_k) \geq 0, \tag{6f}$$

where $\mathbf{f}_i$ and $\mathfrak{g}_i$ denote the dynamics and ground reaction force functions in each mode. Equation (6b) represents the dynamics constraints, (6c) the ground reaction force constraint, and (6d) the impact reset map with $\mathbf{f}_{\text{Imp}}$ the reset map equation. Equation (6e) is a switching constraint requiring the foot to be on the contact surface at any instant of impact. Finally, we group all other constraints, such as torque limits, non-negative normal ground reaction force, and friction constraints, into (6f).

In solving (5), DDP/iLQR provides the optimal control tape as

$$\mathbf{U}_0^\star(\mathbf{x}_0) = \underset{\mathbf{U}_0}{\text{argmin}} \ \mathbf{J}(\mathbf{U}_0),$$

where the superscript $\star$ implies an optimal quantity. Rather than optimizing over the entire control tape, DDP/iLQR solves (5) by optimizing a control policy at each time point, and working backwards in time using Bellman's principle [23]. This process recursively provides a value function approximation as

$$V_k(\mathbf{x}_k) = \min_{\mathbf{u}_k} \left[\underbrace{\ell_k(\mathbf{x}_k, \mathbf{u}_k, \mathfrak{g}(\mathbf{x}_k, \mathbf{u}_k)) + V_{k+1}(\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k))}_{Q_k(\mathbf{x}_k, \mathbf{u}_k)} \right] \tag{7}$$
$$\text{where} \quad V_N(\mathbf{x}_N) = \Phi(\mathbf{x}_N),$$

where the function, $Q_k(\mathbf{x}_k, \mathbf{u}_k)$, captures the cost to go when starting in state $\mathbf{x}_k$ at time k, taking action $\mathbf{u}_k$, and then acting optimally thereafter. Consider the differential change to Q_k in (7) around a nominal state-control pair $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{u}}_k$ with $\delta Q_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) = Q_k(\bar{\mathbf{x}}_k + \delta\mathbf{x}_k, \bar{\mathbf{u}}_k + \delta\mathbf{u}_k) - Q_k(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$, the second-order approximation of the Q function leads to

$$\begin{aligned} Q_{\mathbf{x}} &= \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^\top V'_{\mathbf{x}} + \mathfrak{g}_{\mathbf{x}}^\top \ell_\lambda \\ Q_{\mathbf{u}} &= \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^\top V'_{\mathbf{x}} + \mathfrak{g}_{\mathbf{u}}^\top \ell_\lambda \\ Q_{\mathbf{xx}} &= \ell_{\mathbf{xx}} + \mathbf{f}_{\mathbf{x}}^\top V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + \mathfrak{g}_{\mathbf{x}}^\top \ell_{\lambda\lambda} \mathfrak{g}_{\mathbf{x}} + \begin{bmatrix} V_{\mathbf{x}}'^\top \\ \ell_\lambda^\top \end{bmatrix}^\top \left[\!\left[\begin{matrix} \mathbf{f}_{\mathbf{xx}} \\ \mathfrak{g}_{\mathbf{xx}} \end{matrix} \right]\!\right] \\ Q_{\mathbf{uu}} &= \ell_{\mathbf{uu}} + \mathbf{f}_{\mathbf{u}}^\top V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{u}} + \mathfrak{g}_{\mathbf{u}}^\top \ell_{\lambda\lambda} \mathfrak{g}_{\mathbf{u}} + \begin{bmatrix} V_{\mathbf{x}}'^\top \\ \ell_\lambda^\top \end{bmatrix}^\top \left[\!\left[\begin{matrix} \mathbf{f}_{\mathbf{uu}} \\ \mathfrak{g}_{\mathbf{uu}} \end{matrix} \right]\!\right] \\ Q_{\mathbf{ux}} &= \ell_{\mathbf{ux}} + \mathbf{f}_{\mathbf{u}}^\top V'_{\mathbf{xx}} \mathbf{f}_{\mathbf{x}} + \mathfrak{g}_{\mathbf{u}}^\top \ell_{\lambda\lambda} \mathfrak{g}_{\mathbf{x}} + \begin{bmatrix} V_{\mathbf{x}}'^\top \\ \ell_\lambda^\top \end{bmatrix}^\top \left[\!\left[\begin{matrix} \mathbf{f}_{\mathbf{ux}} \\ \mathfrak{g}_{\mathbf{ux}} \end{matrix} \right]\!\right] \end{aligned} \tag{8}$$

The prime in (8) denotes the next time step, i.e., $V'_{\mathbf{xx}} = V_{\mathbf{xx}}(k+1)$, whereas the subscripts indicate partial derivatives. Throughout this text, $[\![\cdot]\!]_r$ denotes a rank r tensor, with a rank three tensor assumed when r is omitted. When the $[\![\cdot]\!]$ terms are ignored above, i.e., when second-order partials of the dynamics are ignored, the resulting algorithm is known as iLQR [8, 22, 24]. In DDP, $[\cdot]^\top [\![\cdot]\!]$ implies a tensor contraction with a vector to a matrix. Later in this development, we develop a framework that directly computes this tensor-vector contraction without ever forming the tensor.

Minimizing (7) over $\delta\mathbf{u}_k$ attains the incremental control

$$\delta\mathbf{u}_k^\star = \underset{\delta\mathbf{u}_k}{\text{argmin}} \ \delta Q_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) \tag{9}$$
$$= -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} - Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}} \delta\mathbf{x}_k. \tag{10}$$

The resulting control from (10) is used to form the quadratic approximation of the value function as

$$\Delta V(k) = \frac{1}{2} Q_{\mathbf{u}}^\top Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}} + \Delta V(k+1)$$

$$V_{\mathbf{x}}(k) = Q_{\mathbf{x}} - Q_{\mathbf{u}}^{\top} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$$
$$V_{\mathbf{xx}}(k) = Q_{\mathbf{xx}} - Q_{\mathbf{xu}} Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}\,,$$

where $\Delta V(\cdot)$ is the expected cost reduction. This process is repeated until a value function approximation is obtained at time $k = 0$, constituting the backward sweep of DDP. Following this backward sweep, a forward sweep proceeds by simulating the system forward under the incremental control policy (10), resulting in a new state-control trajectory [10]. The sweeps are repeated to convergence.

For the impact event (6d), we consider the quadratic value function approximation across the impact using similar formulas as in [14] that represent a natural generalization of (8). The ground penetration constraint (6d) is considered using an Augmented Lagrangian (AL) approach from [14, 25], whereas other constraints such as torque limits, friction cone constraints, etc. (6f) use a Relaxed Barrier (ReB) method [9, 14, 26]. We refer the interested reader to [14] for detailed derivations and implementation details of how these methods work together.

3 Reducing Complexity in Hybrid Systems DDP

Within DDP/iLQR, the computation of the dynamics partials are the most time intensive operations. We present an approach for reducing the computational complexity of evaluating the tensorial parts of the $Q-$coefficients in (8). We first review conventional methods of computing partials for DDP, and then set the stage for our contributions of computing them efficiently.

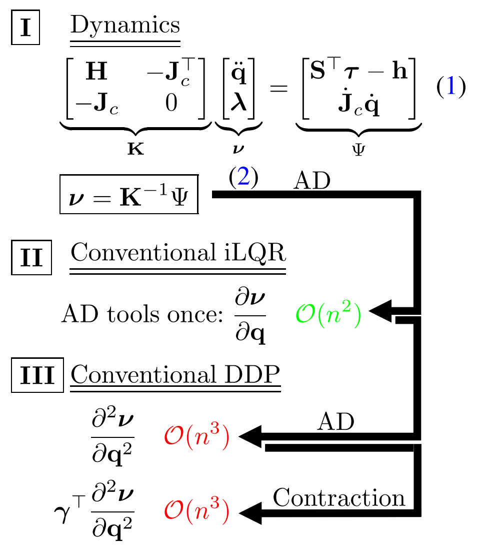
3.1 Conventional Partials

Within DDP, we need both the first- and second-order partial derivatives of (2). Let $\mathbf{z}$ and $\mathbf{y}$ be vectors representing $\mathbf{q}, \dot{\mathbf{q}}$, or τ such that the first partials of the dynamics, $\mathbf{K}\nu = \Psi$ w.r.t. $\mathbf{z}$ are given by:

$$\left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{z}}\right]\!\right] \nu + \mathbf{K}\frac{\partial \nu}{\partial \mathbf{z}} = \frac{\partial \Psi}{\partial \mathbf{z}}\,,$$
$$\therefore \boxed{\frac{\partial \nu}{\partial \mathbf{z}} = \mathbf{K}^{-1}\left[\frac{\partial \Psi}{\partial \mathbf{z}} - \left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{z}}\right]\!\right]\nu\right]}\,. \tag{11}$$

When only the first-order partials (11) are included within the DDP algorithm, the resulting approach is known as *iLQR*. While we note that the functional form of (11) will prove important later on in this development, in implementation, AD tools (e.g., [27]) can be used once on any algorithm that computes $\nu \triangleq \mathbf{K}^{-1}\Psi$, to achieve the desired result $\frac{\partial \nu}{\partial \mathbf{z}}$. For a fixed number of contacts, this operation can be implemented to have $\mathcal{O}(n^2)$ complexity. This complexity is the lowest possible since the operation provides the partials of the n entries of ν w.r.t. the n entries of the $\mathbf{z}$ vector. This approach is illustrated in Fig. 1.

The second-order partials of $\mathbf{K}\nu = \Psi$ for any combination of $\mathbf{z}$ and $\mathbf{y}$ are given as



I
Dynamics
$\underbrace{\begin{bmatrix}\mathbf{H} & -\mathbf{J}_c^\top \\ -\mathbf{J}_c & 0\end{bmatrix}}_{\mathbf{K}}\underbrace{\begin{bmatrix}\ddot{\mathbf{q}} \\ \boldsymbol{\lambda}\end{bmatrix}}_{\boldsymbol{\nu}} = \underbrace{\begin{bmatrix}\mathbf{S}^\top\boldsymbol{\tau} - \mathbf{h} \\ \dot{\mathbf{J}}_c\dot{\mathbf{q}}\end{bmatrix}}_{\Psi}$ (1)
$\boldsymbol{\nu} = \mathbf{K}^{-1}\Psi$
(2)
AD
II
Conventional iLQR
AD tools once: $\frac{\partial \boldsymbol{\nu}}{\partial \mathbf{q}}$
$\mathcal{O}(n^2)$
III
Conventional DDP
$\frac{\partial^2 \boldsymbol{\nu}}{\partial \mathbf{q}^2}$
$\mathcal{O}(n^3)$
AD
$\gamma^\top \frac{\partial^2 \boldsymbol{\nu}}{\partial \mathbf{q}^2}$
$\mathcal{O}(n^3)$
Contraction

Fig. 1: (I) The dynamics function (II) Conventional iLQR, and (III) Conventional DDP approach.

$$\left[\!\left[\frac{\partial^2 \mathbf{K}}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right]_4 \nu + \left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{z}}\right]\!\right]\frac{\partial \nu}{\partial \mathbf{y}} +$$
$$\left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{y}}\right]\!\right]\frac{\partial \nu}{\partial \mathbf{z}} + \mathbf{K}\left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right] = \left[\!\left[\frac{\partial^2 \Psi}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right] \tag{12}$$
$$\therefore \boxed{\left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right] = \mathbf{K}^{-1}\left[\left[\!\left[\frac{\partial^2 \Psi}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right] - \left[\!\left[\frac{\partial^2 \mathbf{K}}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right]_4 \nu - (\mathcal{A}+\mathcal{B})\right]}$$
$$\text{where}\quad \mathcal{A} = \left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{y}}\frac{\partial \nu}{\partial \mathbf{z}}\right]\!\right] \text{ and } \mathcal{B} = \left[\!\left[\frac{\partial \mathbf{K}}{\partial \mathbf{z}}\frac{\partial \nu}{\partial \mathbf{y}}\right]\!\right]\,.$$

Note that $\mathcal{A} = 0$ when $\mathbf{y} \neq \mathbf{q}$ and $\mathcal{B} = 0$ when $\mathbf{z} \neq \mathbf{q}$. The result of (12) is a rank three tensor. For example, $\left[\!\left[\frac{\partial^2 \Psi}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right]$ considers the n^2 partials of the n entries of Ψ due to the n entries of $\mathbf{z}$ and $\mathbf{y}$. The term $\left[\!\left[\frac{\partial^2 \mathbf{K}}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right]_4$ is a rank four tensor contracted by a vector ν to attain a rank three tensor.

Remark 1. *As formulated,* (12) *involves several matrix-tensor, tensor-vector, and tensor transpose operations. Their implementation details are left out in this letter, though we acknowledge that most operations require careful convention choices for the chain rule operations to provide correct results.*

For implementation, AD tools can be used twice on $\nu \triangleq \mathbf{K}^{-1}\Psi$ to achieve the desired results. In using AD tools twice, computing $\left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{z}\partial \mathbf{y}}\right]\!\right]$ is an $\mathcal{O}(n^3)$ operation to obtain the partials of the n entries of ν w.r.t. the n entries of the $\mathbf{z}$ and the n entries of the $\mathbf{y}$ vectors. This approach is illustrated in Fig. 1. Within the DDP context, the second-order partials will be contracted with the fixed vector $\gamma^\top \triangleq [V_{\mathbf{x}}'^\top \; \ell_\lambda^\top]^\top$ (see (8)) from the left. The tensor-contraction operation is also an $\mathcal{O}(n^3)$ operation. Since this AD approach requires forming a tensor operator, whenever we use this method, the resulting approach is denoted *Tensor DDP*. We aim to reduce the $\mathcal{O}(n^3)$ complexity of these second-order partials by exploiting their structure.

3.2 Proposed Refactoring for Efficient Computation

Alternatively, consider the contraction of the second-order partials (12) with a fixed vector $\gamma^\top$ as needed for DDP:

$$\gamma^\top \left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{z} \partial \mathbf{y}}\right]\!\right] = \gamma^\top \mathbf{K}^{-1}\left(\left[\!\left[\frac{\partial^2 \Psi}{\partial \mathbf{z} \partial \mathbf{y}}\right]\!\right] - \left[\!\left[\frac{\partial^2 \mathbf{K}}{\partial \mathbf{z} \partial \mathbf{y}}\right]\!\right]_4 \nu - (\mathcal{A}+\mathcal{B})\right). \quad (13)$$

Here, as we do not need partials of $\gamma^\top \mathbf{K}^{-1}$, we consider it as a fixed vector (i.e., that its partials w.r.t. $\mathbf{z}$ or $\mathbf{y}$ are zero). We define $\xi \triangleq -\mathbf{K}^{-1}\gamma$. For the remainder of this development, we focus on the second-order partials w.r.t. $\mathbf{z}=\mathbf{y}=\mathbf{q}$. We note that for these $\mathbf{q}$ partials, $\mathcal{B}=\mathcal{A}^\top$ and we rewrite (12) as:

$$\gamma^\top \left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{q} \partial \mathbf{q}}\right]\!\right] = -\xi^\top \left[\!\left[\frac{\partial^2 \Psi}{\partial \mathbf{q} \partial \mathbf{q}}\right]\!\right] + \xi^\top \left[\!\left[\frac{\partial^2 \mathbf{K}}{\partial \mathbf{q} \partial \mathbf{q}}\right]\!\right]_4 \nu + \xi^\top(\mathcal{A}+\mathcal{A}^\top), \quad (14)$$

where $\xi^\top$ contracts each rank three tensor term to a matrix.

Rather than lumping several quantities into the $\mathbf{K}, \nu$ and Ψ terms in (14), we separate the functions into their constituent terms and partition $\xi = [\mu^\top \quad \pi^\top]^\top$, where $\mu \in \mathbb{R}^n$ and $\pi \in \mathbb{R}^{n_c}$. This enables us to rewrite (14) as:

$$\gamma^\top \left[\!\left[\frac{\partial^2 \nu}{\partial \mathbf{q} \partial \mathbf{q}}\right]\!\right] = \mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_2^\top, \text{ where} \quad (15)$$

$$\mathbf{T}_1 = \mu^\top \left[\!\left[\frac{\partial^2}{\partial \mathbf{q}^2}\left[\mathbf{H}\ddot{\mathbf{q}} + \mathbf{h} - \mathbf{J}_c^\top \lambda\right]\right]\!\right] - \pi^\top \left[\!\left[\frac{\partial^2}{\partial \mathbf{q}^2}\left[\dot{\mathbf{J}}_c\dot{\mathbf{q}} + \mathbf{J}_c\ddot{\mathbf{q}}\right]\right]\!\right]$$

$$\mathbf{T}_2 = \mu^\top \left[\!\left[\frac{\partial \mathbf{H}}{\partial \mathbf{q}}\frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}} - \frac{\partial \mathbf{J}_c^\top}{\partial \mathbf{q}}\frac{\partial \lambda}{\partial \mathbf{q}}\right]\!\right] - \pi^\top \left[\!\left[\frac{\partial \mathbf{J}_c}{\partial \mathbf{q}}\frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}}\right]\!\right].$$

This new structural form for the second-order derivatives of contact-constrained dynamics represents the first contribution of this letter. The corresponding second-order partials w.r.t. $\mathbf{q}, \dot{\mathbf{q}}$, and τ take the same general form as (15) and their details are included in Appendix A.

4 Efficient Partials via Modifications to the RNEA

The structural form of $\mathbf{T}_1$ in (15) points toward the development of a variant of the RNEA that we call the modified RNEA for contacts (mRNEAc), which provides:

$$\begin{aligned}\mu^\top \tau - \pi^\top \boldsymbol{a}_c &= \text{mRNEAc}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \lambda, \mu, \pi) \\ &= \mu^\top\left[\mathbf{H}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}_c^\top \lambda\right] \\ &\quad - \pi^\top\left[\mathbf{J}_c\ddot{\mathbf{q}} + \dot{\mathbf{J}}_c\dot{\mathbf{q}}\right],\end{aligned} \quad (16)$$

where $\boldsymbol{a}_c = \mathbf{J}_c\ddot{\mathbf{q}} + \dot{\mathbf{J}}_c\dot{\mathbf{q}}$ gives the contact acceleration. Both inputs μ and π are considered as fixed vectors (i.e., such that taking their partials w.r.t. $\mathbf{q}$ is zero). This definition (16) is motivated by the fact that then

$$\mathbf{T}_1 = \frac{\partial}{\partial \mathbf{q}}\left[\frac{\partial}{\partial \mathbf{q}} \text{mRNEAc}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \lambda, \mu, \pi)\right]. \quad (17)$$

Algorithm 1 Modified RNEA Algorithm for Contacts (mRNEAc)

```
Require: model, $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \lambda, \mu, \pi$
 1: $\mathbf{v}_0 = 0, \mathbf{a}_0 = 0, \mathbf{a}_{g_0} = \mathbf{a}_g, \mathbf{w}_0 = 0, s = 0$
 2: for $i = 1$ to $N$ do
 3:     $\mathbf{v}_i = {}^i\mathbf{X}_{p(i)}\,\mathbf{v}_{p(i)} + \mathbf{S}_i\,\dot{\mathbf{q}}_i$
 4:     $\mathbf{w}_i = {}^i\mathbf{X}_{p(i)}\,\mathbf{w}_{p(i)} + \mathbf{S}_i\,\mu_i$
 5:     $\mathbf{a}_{g_i} = {}^i\mathbf{X}_{p(i)}\,\mathbf{a}_{g_{p(i)}}$
 6:     $\mathbf{a}_i = {}^i\mathbf{X}_{p(i)}\,\mathbf{a}_{p(i)} + (\mathbf{v}_i\times)\,\mathbf{S}_i\dot{\mathbf{q}}_i + \mathbf{S}_i\ddot{\mathbf{q}}_i$
 7:     $\mathbf{F}_i = \mathbf{I}_i(\mathbf{a}_i - \mathbf{a}_{g_i}) + (\mathbf{v}_i\times^*)\,\mathbf{I}_i\mathbf{v}_i - \lambda_i$
 8:     $s += \mathbf{w}_i^\top \mathbf{F}_i - \pi_i^\top \mathbf{a}_i$
 9: end for
10: return $[s = \mu^T \tau - \pi^\top \boldsymbol{a}_c]$
```

Further, consider quantities $\frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}}$ and $\frac{\partial \lambda}{\partial \mathbf{q}}$ that are provided as fixed matrices. With these fixed matrices as inputs to the mRNEAc, and taking the partials of the resulting output, we also obtain that

$$\mathbf{T}_2 = \frac{\partial}{\partial \mathbf{q}} \text{mRNEAc}\left(\mathbf{q}, 0, \frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}}, 0, \frac{\partial \lambda}{\partial \mathbf{q}}, \mu, \pi\right). \quad (18)$$

Note that the outputs of the mRNEAc require similar components as the RNEA, but with the inclusion of the effects of contacts forces, $\mathbf{J}_c^\top \lambda$, and the contact acceleration, $\dot{\mathbf{J}}_c\dot{\mathbf{q}} + \mathbf{J}_c\ddot{\mathbf{q}}$. In [19, Algorithm 1], a modified RNEA (mRNEA) was introduced that outputs $\mu^\top \tau$. The mRNEAc is an extension of that algorithm to include contact effects, and is presented in Algo. 1 using Featherstone's rigid-body dynamics notation [21][1]. The mRNEAc is an $\mathcal{O}(n)$ one-pass algorithm and thus has a simpler computation graph compared to RNEA. This development represents the second technical contribution in this letter.

Additionally, this $\mathcal{O}(n)$ algorithm enables the use of Reverse-mode AD [28] tools. Reverse AD provides a general-purpose approach to compute the gradient of any scalar-valued function in the same complexity as the evaluation of the function itself. This result is also known as the "cheap gradient principle", where the computational cost of evaluating reverse AD is bounded above by a small constant (at most 5) times the cost of evaluating the function itself [28].

The result is that by using the mRNEAc with reverse AD, we can efficiently compute all the terms in (15) without resorting to tensor operations, and in lower computational complexity than tensor DDP. As a key outcome, the term $\mathbf{T}_1$ can be calculated in $\mathcal{O}(n^2)$ complexity. First, reverse-mode AD can be used to compute $\frac{\partial}{\partial \mathbf{q}}\text{mRNEAc}(\cdot)$ in $\mathcal{O}(n)$. Then, AD can be used again for the partial of that result, setting the complexity at $\mathcal{O}(n^2)$. The term $\mathbf{T}_2$ can also be calculated in $\mathcal{O}(n^2)$ complexity, but by running mRNEAc n times with the n columns of $\frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}}$ as input. Taking the partials

[1]There are a few subtleties to make use of λ and π in the algorithm, which are discussed in Appendix B.

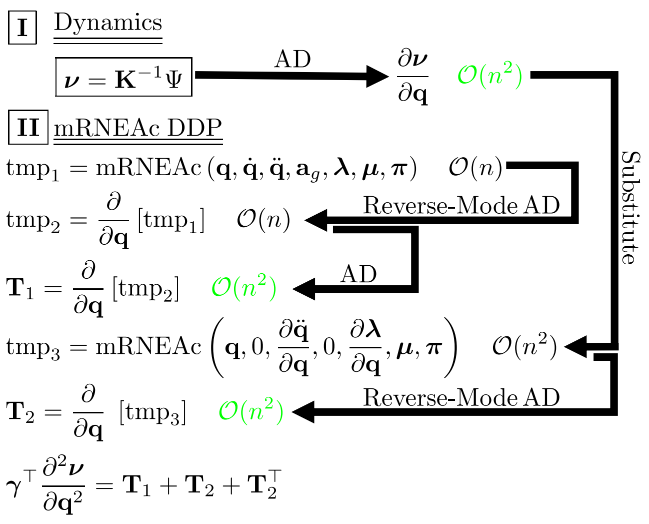

I Dynamics
$\nu = \mathbf{K}^{-1}\Psi$
AD
$\frac{\partial \nu}{\partial \mathbf{q}}$
$\mathcal{O}(n^2)$
II mRNEAc DDP
$\mathrm{tmp}_1 = \mathrm{mRNEAc}\left(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\pi}\right)$ $\mathcal{O}(n)$
$\mathrm{tmp}_2 = \frac{\partial}{\partial \mathbf{q}}[\mathrm{tmp}_1]$ $\mathcal{O}(n)$
Reverse-Mode AD
$\mathbf{T}_1 = \frac{\partial}{\partial \mathbf{q}}[\mathrm{tmp}_2]$ $\mathcal{O}(n^2)$
AD
$\mathrm{tmp}_3 = \mathrm{mRNEAc}\left(\mathbf{q}, 0, \frac{\partial \ddot{\mathbf{q}}}{\partial \mathbf{q}}, 0, \frac{\partial \boldsymbol{\lambda}}{\partial \mathbf{q}}, \boldsymbol{\mu}, \boldsymbol{\pi}\right)$ $\mathcal{O}(n^2)$
$\mathbf{T}_2 = \frac{\partial}{\partial \mathbf{q}}[\mathrm{tmp}_3]$ $\mathcal{O}(n^2)$
Reverse-Mode AD
Substitute
$\gamma^\top \frac{\partial^2 \nu}{\partial \mathbf{q}^2} = \mathbf{T}_1 + \mathbf{T}_2 + \mathbf{T}_2^\top$

Fig. 2: (I) The dynamics function and (II) the proposed mRNEAc DDP approach for the second-order partials of $\gamma^\top \frac{\partial^2 \nu}{\partial \mathbf{q}^2}$. Details of the other second-order partials are included in the Appendix A.

of that result using AD sets the complexity of this operation at $\mathcal{O}(n^2)$. This approach is illustrated in Fig. 2.

The effect is that all the second-order partials are calculated efficiently in $\mathcal{O}(n^2)$ and without requiring any tensor operations. Overall, this approach reduces the computational complexity of taking the second-order partials needed in DDP as compared to applying tensor contraction. This development represents the third technical contribution in this letter, with this approach for DDP termed *mRNEAc DDP*.

5 Results

Dynamics Partials

First, we evaluate the proposed methods on an underactuated $n-$link pendubot model (Fig. 3) whose last link is pinned to the ground through a revolute joint. The $n-$link pendubot model allows us to test the scalability of the proposed methods with an increasing number of DoFs. Our analysis was carried out using MATLAB with CasADi [27], which allows for rapid and efficient testing of AD approaches. We first compare the computation time of the different methods for evaluating partials for iLQR/DDP.

The computation time for evaluating the dynamics partials is shown in Fig. 4, where each point represents the

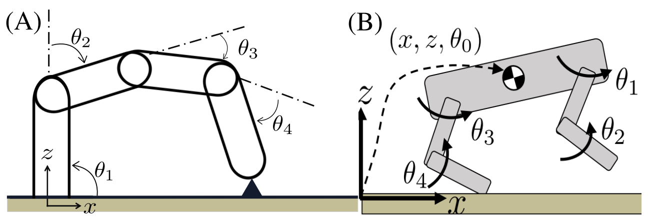


Fig. 3: (A) The pinned pendubot model and (B) the planar quadruped model.

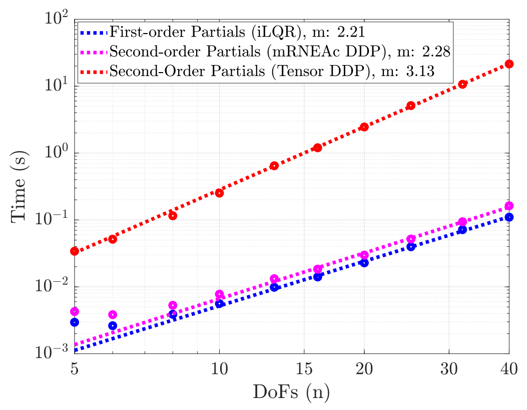


Fig. 4: The computation time of the dynamics partials using the proposed methods for the pinned pendubot model. The slope of each plot is also illustrated.

average time over ten calls. Figure 4 is presented on a log-log scale, and as such, any curve of the form $y = Cx^p$ will appear as a line $\log(y) = \log(C) + p\log(x)$ whose slope corresponds to the polynomial power. As such, the slope of the plotted lines represents the empirically observed polynomial order of the algorithm. The evaluation of the second-order partials using Tensor DDP took the longest time. The computation of the second-order partials for mRNEAc DDP (slope of 2.28) showed a order reduction compared to Tensor DDP (slope of 3.13). As shown, the time to compute the second-order partials for mRNEAc DDP was a comparable order of magnitude as the first-order partials of iLQR (slope of 2.21). Here, the computational cost of calculating the second-order partials for mRNEAc DDP was at most 1.39 times the cost of evaluating first-order partials for iLQR. We note, however, that compared to first-order partials, second-order partials involve proportionally more mathematical operations. As such, the computation of the second-order partials for mRNEAc DDP takes slightly more time compared to first-order partials for iLQR. The computation time benefits for DDP were especially more apparent for systems with a higher number of DoFs, indicating the potential of the proposed methods to include second-order information into TO algorithms for legged robots or other high-DoF systems. For systems with a lower number of DoFs, the computation time is dominated by overhead, and there is no marked difference in the computation time of the methods.

Trajectory Optimization

Further, we consider TO for a bounding gait with a planar quadruped - a 2D model of the MIT mini cheetah [29]. The model of the system (see Fig. 3) has 5 links (torso, and 2 links per leg) and has $n = 7$ DoFs. The bounding sequence is encoded via a design of running and terminal costs in each mode that aim to track a reference configuration, and are given as

$$\ell_i(\mathbf{x}, \mathbf{u}, \lambda) = \left[\mathbf{x} - \mathbf{x}_{\mathrm{ref},i}\right]^\top \mathbf{Q}_i \left[\mathbf{x} - \mathbf{x}_{\mathrm{ref},i}\right] + \mathbf{u}^\top \mathbf{R}_i \mathbf{u} + \left[\lambda - \lambda_{\mathrm{ref,i}}\right]^\top \mathbf{S}_i \left[\lambda - \lambda_{\mathrm{ref,i}}\right],$$

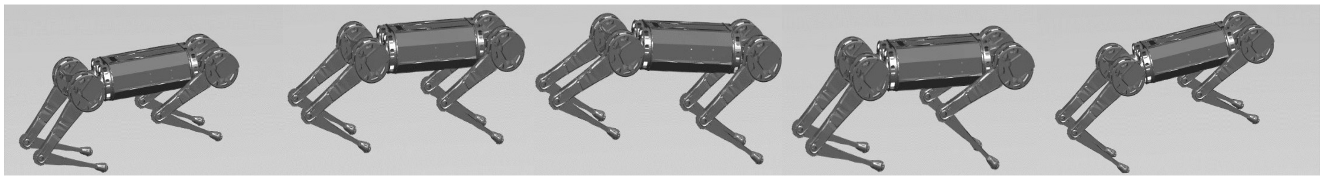


Fig. 5: Snapshots of the resulting optimized motion by iLQR/DDP.

$$\Phi_i(\mathbf{x}_{N_{i+1}^-}) = [\mathbf{x} - \mathbf{x}_{\text{ref},i}]^\top \mathbf{Q}_{fi} [\mathbf{x} - \mathbf{x}_{\text{ref},i}] .$$

Here, $\mathbf{x}_{\text{ref},i}$ refers to a pre-defined reference configuration in mode i and $\lambda_{\text{ref},i}$ refers to reference ground reaction forces. The terms $\mathbf{Q}, \mathbf{R}$, and $\mathbf{S}$ are weighting matrices for the state, control, and contact forces, respectively, in each mode. As such, this formulation aims to attain a pre-defined configuration while reducing energy consumption and attaining the requisite ground reaction forces. The state weighting matrix, $\mathbf{Q}_{fi}$, in the terminal cost penalizes deviations that the robot should attain in its final state in each mode.

The resulting motion following the TO by iLQR/DDP is illustrated in Fig. 5. The robot starts in the back-stance mode and runs forward at a speed of 0.5m/s. The initial guess trajectory for this motion was constructed via a heuristic controller that implements PD control in the flight mode to achieve a predefined configuration. In the stance mode, the heuristic controller calculates stance leg forces following a SLIP model and converts the obtained Ground Reaction Forces (GRF) to joint torques. The initial states assume the robot to be moving forward at 0.5m/s. The hybrid systems iLQR/DDP algorithm [14] considers ground penetration, torque limits, non-negative normal ground reaction force, and friction cone constraints. The resulting motion respects those constraints as shown in Fig. 6 with a snapshot of the final motion included in Fig. 5.

Further, we consider the timing demands of the iLQR/DDP approaches. We consider 50 different initial trajectories, where the iLQR and the DDP variants were optimized for 50 iterations. As shown in Fig. 7, iLQR on average took the least time as compared to the DDP variants, whereas Tensor DDP took the most time. As compared to iLQR, on average, mRNEAc DDP took approximately 3.32 times more computation time than iLQR, whereas Tensor DDP took 28.3 times more computation time. As such, the computational cost of mRNEAc DDP was bounded above, on average, by 3.32 times the cost of evaluating iLQR. The evaluation of the dynamics partials (see Fig. 4) via the full mRNEAc DDP took comparatively more time over iLQR since DDP has to first compute the first-order partials and then the second-order partials. Further, DDP often requires a regularization scheme [8] and may occasionally repeat the backward pass to ensure effective cost reduction, which requires additional time.

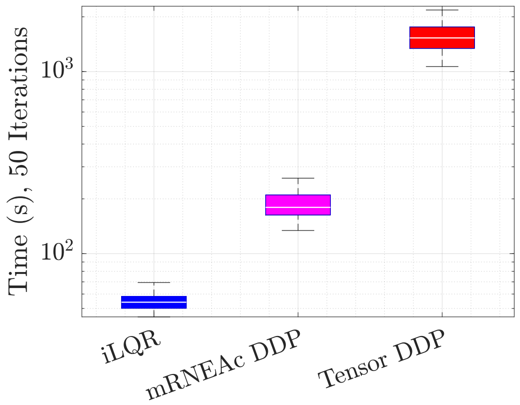


Fig. 7: Timing comparison of the proposed methods as compared to the conventional methods.

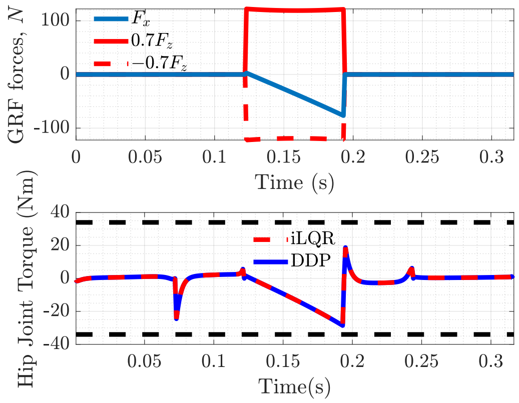


Fig. 6: (a) Ground Reaction forces satisfying the friction cone constraint for both iLQR/DDP solutions. (b) Joint torque control satisfying the control limits. DDP and iLQR converged to the same solution.

6 Discussion

The proposed DDP approach provides a reduction in the computational complexity and computation time as compared to the conventional DDP approach. The computational complexity of our method for computing second-order derivatives in DDP was the same as computing first-order derivatives for iLQR. Further, full second-order DDP offers quadratic convergence, as compared to iLQR's superlinear convergence, from near-optimal trajectories. As a result, convergence with DDP can often be achieved in less iterations. Further, DDP retains better local fidelity to the dynamics model and better captures the nonlinear effects of the model. As such, for a robot facing modeling inaccuracy (e.g., slight terrain variation) in its contacts with the environment, the increased local fidelity of the robot model could help prevent falls. Moreover, in unstructured environments, when terrain information is known, reasoning about the full model with our methods could help the robot more quickly tailor its trajectories to the environment thanks to DDP's quadratic convergence.

7 Conclusions and Future Directions

This paper presented several advances that accelerate the inclusion of second-order information into DDP for application to systems making rigid contact with the environment. This work leveraged reverse-mode AD tools and a refactoring of RNEA known as mRNEAc to efficiently compute second-order partials while avoiding a costly vector-tensor contraction operation. The result presented is a DDP approach that computes second-order partials in the same complexity as first-order partials in iLQR. As compared to conventional DDP, the proposed DDP greatly reduces the computation overhead.

Several opportunities motivate the next steps of this work. First, for conciseness sake, Forward Euler was used as the integration scheme, though we also see an opportunity to extend this work to implicit [30, 31] and predictor-corrector integration schemes to further improve numerical integration stability. Second, this work focused on the technical derivations of our contributions and we see value in validating this work on legged robot hardware, which would represent the first use-case of full DDP on a legged robot. Third, while this work considers a single-shooting DDP framework, several methods in the literature [16, 32] have shown the potential of a multi-shooting framework to reduce TO's sensitivity to initial conditions. We see the potential to use the same approaches as proposed herein within a multi-shooting framework. Finally, we note that DDP has in theory been shown to exhibit quadratic convergence from near-optimal trajectories [10]. As such, we consider an opportunity for TO to exploit that convergence property with robots operating in uncertain and varied terrains. We also consider theoretical underpinnings on whether that property holds for hybrid systems DDP as [10] only developed theoretical convergence guarantees for smooth dynamics.

References

[1] Apgar, T., Clary, P., Green, K., Fern, A., and Hurst, J. W., 2018. "Fast online trajectory optimization for the bipedal robot cassie.". In Robotics: Science and Systems, Vol. 101, p. 14.

[2] Wensing, P. M., and Orin, D. E., 2013. "High-speed humanoid running through control with a 3D-SLIP model". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5134–5140.

[3] Koditschek, D. E., and Full, R. J., 1999. "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land". *Journal of Experimental Biology,* **202**(23), pp. 3325–3332.

[4] Hutter, M., Remy, C. D., Höpflinger, M. A., and Siegwart, R., 2010. "SLIP running with an articulated robotic leg". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4934–4939.

[5] Dai, H., Valenzuela, A., and Tedrake, R., 2014. "Whole-body motion planning with centroidal dynamics and full kinematics". In IEEE-RAS International Conference on Humanoid Robots, pp. 295–302.

[6] Bledt, G., Powell, M. J., Katz, B., Di Carlo, J., Wensing, P. M., and Kim, S., 2018. "MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2245–2252.

[7] Bellicoso, C. D., Jenelten, F., Gehring, C., and Hutter, M., 2018. "Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots". *IEEE Robotics and Automation Letters,* **3**(3), pp. 2261–2268.

[8] Tassa, Y., Erez, T., and Todorov, E., 2012. "Synthesis and stabilization of complex behaviors through online trajectory optimization". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4906–4913.

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

[10] Mayne, D., 1966. "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems". *International Journal of Control,* **3**(1), pp. 85–95.

[11] Tassa, Y., Mansard, N., and Todorov, E., 2014. "Control-limited differential dynamic programming". In IEEE International Conference on Robotics and Automation, pp. 1168–1175.

[12] Lantoine, G., and Russell, R. P., 2012. "A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: Theory". *Journal of Optimization Theory and Applications,* **154**(2), pp. 382–417.

[13] Howell, T., Jackson, B., and Manchester, Z., 2019. "Altro: A fast solver for constrained trajectory optimization". In IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 7674 – 7679.

[14] Li, H., and Wensing, P. M., 2020. "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots". *IEEE Robotics and Automation Letters,* **5**(4), pp. 5448–5455.

[15] Kong, N. J., Council, G., and Johnson, A. M., 2021. "iLQR for piecewise-smooth hybrid dynamical systems". In IEEE Conference on Decision and Control, pp. 5374–5381.

[16] Tang, Y., Chu, X., and Au, K., 2021. "HM-DDP: A hybrid multiple-shooting differential dynamic programming method for constrained trajectory optimization". *arXiv preprint arXiv:2109.07131*.

[17] Li, H., Frei, R. J., and Wensing, P. M., 2021. "Model hierarchy predictive control of robotic systems". *IEEE Robotics and Automation Letters,* **6**(2), pp. 3373–3380.

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

[19] Nganga, J., and Wensing, P. M., 2021. "Accelerating second-order differential dynamic programming for rigid-body systems". *IEEE Robotics and Automation Letters,* **6**(4), pp. 7659–7666.

[20] Orin, D. E., McGhee, R., Vukobratović, M., and Hartoch, G., 1979. "Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods". *Math Biosciences,* **43**(1-2), pp. 107–130.

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

[22] Budhiraja, R., 2019. "Multi-body locomotion: Problem structure and efficient resolution". PhD thesis, Institut national des sciences appliquées de Toulouse.

[23] Denardo, E. V., 2012. *Dynamic programming: models and applications*. Courier Corporation.

[24] Budhiraja, R., Carpentier, J., Mastalli, C., and Mansard, N., 2018. "Differential dynamic programming for multi-phase rigid contact dynamics". In IEEE-RAS Int. Conf. on Humanoid Robots, pp. 1–9.

[25] Farshidian, F., Neunert, M., Winkler, A. W., Rey, G., and Buchli, J., 2017. "An efficient optimal planning and control framework for quadrupedal locomotion". In IEEE International Conference on Robotics and Automation, pp. 93–100.

[26] Hauser, J., and Saccon, A., 2006. "A barrier function method for the optimization of trajectory functionals with constraints". In IEEE Conference on Decision and Control, pp. 864–869.

[27] Andersson, J. A., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M., 2019. "CasADi: a software framework for nonlinear optimization and optimal control". *Mathematical Programming Computation,* **11**(1), pp. 1–36.

[28] Grievank, A., 2000. "Principles and techniques of algorithmic differentiation: Evaluating derivatives". *SIAM, Philadelphia*.

[29] Katz, B., Di Carlo, J., and Kim, S., 2019. "Mini cheetah: A platform for pushing the limits of dynamic quadruped control". In IEEE International Conference on Robotics and Automation, pp. 6295–6301.

[30] Chatzinikolaidis, I., and Li, Z., 2021. "Trajectory optimization of contact-rich motions using implicit differential dynamic programming". *IEEE Robotics and Automation Letters,* **6**(2), pp. 2626–2633.

[31] Jallet, W., Mansard, N., and Carpentier, J., 2022. "Implicit differential dynamic programming". In International Conference on Robotics and Automation, pp. 1455–1461.

[32] Pellegrini, E., and Russell, R. P., 2017. "Applications of the multiple-shooting differential dynamic programming algorithm with path and terminal constraints". In AAS/AIAA Astrodynamics Specialist Conference, Stevenson, WA.

A All Second-order Partials

This appendix provides the definitions of the partials of $\frac{\partial^2 \nu}{\partial \mathbf{z} \partial \mathbf{y}}$ where $\mathbf{z}$ and $\mathbf{y}$ can be $\mathbf{q}$, $\dot{\mathbf{q}}$, or τ. The derivation of $\frac{\partial^2 \nu}{\partial \mathbf{q}^2}$ has been provided in (15), (17), and (18). Derivations for the other second-order partials follow similarly. We include the results of all nonzero second-order partials below.

Second-order partials w.r.t. $\mathbf{q}, \dot{\mathbf{q}}$:

$$\left[\gamma^\top\right] \frac{\partial^2 \nu}{\partial \mathbf{q} \partial \dot{\mathbf{q}}} = \mathbf{T}_3 + \mathbf{T}_4 \quad \text{where}$$

$$\mathbf{T}_3 = \frac{\partial}{\partial \mathbf{q}} \left[\frac{\partial}{\partial \dot{\mathbf{q}}} \ \text{mRNEAc}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \lambda, \mu, \pi) \right]$$

$$\mathbf{T}_4 = \frac{\partial}{\partial \mathbf{q}} \ \text{mRNEAc}\left(\mathbf{q}, 0, \frac{\partial \ddot{\mathbf{q}}}{\partial \dot{\mathbf{q}}}, 0, \frac{\partial \lambda}{\partial \dot{\mathbf{q}}}, \mu, \pi \right)$$

Second-order partials w.r.t. $\dot{\mathbf{q}}, \dot{\mathbf{q}}$:

$$\left[\gamma^\top\right] \frac{\partial^2 \nu}{\partial \dot{\mathbf{q}} \partial \dot{\mathbf{q}}} = \mathbf{T}_5 + \mathbf{T}_6 \quad \text{where}$$

$$\mathbf{T}_5 = \frac{\partial}{\partial \dot{\mathbf{q}}} \left[\frac{\partial}{\partial \dot{\mathbf{q}}} \ \text{mRNEAc}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{a}_g, \lambda, \mu, \pi) \right]$$

$$\mathbf{T}_6 = \frac{\partial}{\partial \dot{\mathbf{q}}} \ \text{mRNEAc}\left(\mathbf{q}, 0, \frac{\partial \ddot{\mathbf{q}}}{\partial \dot{\mathbf{q}}}, 0, \frac{\partial \lambda}{\partial \dot{\mathbf{q}}}, \mu, \pi \right)$$

Second-order partials w.r.t. $\mathbf{q}, \tau$:

$$\left[\gamma^\top\right] \frac{\partial^2 \nu}{\partial \mathbf{q} \partial \tau} = \mathbf{T}_7 \quad \text{where}$$

$$\mathbf{T}_7 = \frac{\partial}{\partial \mathbf{q}} \ \text{mRNEAc}\left(\mathbf{q}, 0, \frac{\partial \ddot{\mathbf{q}}}{\partial \tau}, 0, \frac{\partial \lambda}{\partial \tau}, \mu, \pi \right)$$

B Details of π and λ Transformations

In this appendix, we discuss how to make proper use of λ and π within the mRNEAc algorithm. For simplicity, we consider the case of a single point-contact constraint, while the development cleanly generalizes to other contact scenarios (e.g., line contact, planar contact, etc.). In (14), we had that $\xi = [\mu^\top \ \ \pi^\top]^\top$. Looking at $\mathbf{T}_1$ in (15), we observe that $\pi \in \mathbb{R}^{n_c}$ (where $n_c = 3$ for point contacts) pre-multiplies the Cartesian contact acceleration $\boldsymbol{a}_c = \dot{\mathbf{J}}_c \dot{\mathbf{q}} + \mathbf{J}_c \ddot{\mathbf{q}}$. Further, we note that the calculations within the RNEA determine spatial (i.e., 6D) velocities $\mathbf{v}_i$ and accelerations $\mathbf{a}_i = [\dot{\omega}_i, \dot{\boldsymbol{v}}_i]$ of each body i [21], where $\dot{\omega}_i$ is the angular acceleration of a frame attached to the body, and $\dot{\boldsymbol{v}}_i$ the rate of change in the body-frame velocity of its origin.

We can use this information to calculate the acceleration of any contact points. For example, assuming zero contact velocity, the acceleration of contact point c on body i can be calculated by $\boldsymbol{a}_c = \dot{\boldsymbol{v}}_i + \dot{\omega}_i \times \boldsymbol{p}_{c/i}$. We can thus define a matrix $\mathbf{Z} \in \mathbb{R}^{n_c \times 6}$ such that $\boldsymbol{a}_c = \mathbf{Z}\mathbf{a}_i$. If we define the spatial vector $\pi_i = \mathbf{Z}^\top \pi \in \mathbb{R}^6$, we have that

$$\pi^\top [\dot{\mathbf{J}}_c \dot{\mathbf{q}} + \mathbf{J}_c \ddot{\mathbf{q}}] = \pi^\top \boldsymbol{a}_c = \pi^\top \mathbf{Z} \mathbf{a}_i = \pi_i^\top \mathbf{a}_i \, ,$$

where π_i is used in the mRNEAc. Likewise, we can convert contact forces $\lambda \in \mathbb{R}^{n_c}$ to a spatial (6D) force/moment pair via defining $\lambda_i = \mathbf{Z}^\top \lambda \in \mathbb{R}^6$, which is also used in the mRENAc. This construction generalizes readily to multiple contacts, where bodies not in contact are assigned $\pi_i = 0$ and $\lambda_i = 0$ within the mRNEAc. We note that the above transformations can be readily obtained by placing the body-fixed reference frame at the contact point.