

A Unified Perspective on Multiple Shooting In Differential Dynamic Programming

He Li¹, Wenhao Yu², Tingnan Zhang², and Patrick M. Wensing¹

Abstract—Differential Dynamic Programming (DDP) is an efficient computational tool for solving nonlinear optimal control problems. It was originally designed as a single shooting method and thus is sensitive to the initial guess supplied. This work considers the extension of DDP to multiple shooting (MS), improving its robustness to initial guesses. A novel derivation is proposed that accounts for the defect between shooting segments during the DDP backward pass, while still maintaining quadratic convergence locally. The derivation enables unifying multiple previous MS algorithms, and opens the door to many smaller algorithmic improvements. A penalty method is introduced to strategically control the step size, further improving the convergence performance. An adaptive merit function and a more reliable acceptance condition are employed for globalization. The effects of these improvements are benchmarked for trajectory optimization with a quadrotor, an acrobot, and a manipulator. MS-DDP is also demonstrated for use in Model Predictive Control (MPC) for dynamic jumping with a quadruped robot, showing its benefits over a single shooting approach. Video link.

I. INTRODUCTION

Model Predictive Control (MPC) is a powerful technique for controlling complex systems and has been widely used for many robotic systems, including quadrotors [1], quadruped robots [2], [3], and humanoid robots [4]. MPC needs to efficiently and reliably solve a sequence of finite horizon optimal control problems (OCPs) of the form

$$\min_{\mathbf{u}(\cdot)} \int_0^T \ell_c(\mathbf{x}(t), \mathbf{u}(t)) dt + \phi(\mathbf{x}(T)) \quad (1a)$$

$$\text{subject to} \quad \dot{\mathbf{x}} = \mathbf{f}_c(\mathbf{x}, \mathbf{u}) \quad (1b)$$

where T is the prediction horizon, \mathbf{x} the state variable, \mathbf{u} the control variable, ℓ_c the running cost, ϕ the terminal cost, and \mathbf{f}_c the dynamics function. The problem (1) is an infinite-dimensional optimization problem, as it is in continuous time, and the dynamics are highly nonlinear for many robotics systems. Therefore, an analytical solution, in general, does not exist, and numerical methods are often employed. One commonly used class of approaches are *direct methods*. A *direct method* parameterizes the state and control using a finite number of variables, and transcribes the original OCP (1) into a nonlinear optimization problem. The problem transcription most often takes one of three approaches, *single shooting* (SS), *multiple shooting* (MS), and *collocation*. This work focuses on the first two approaches.

¹ He Li and Patrick M. Wensing are with the University of Notre Dame, IN, USA (hli25@nd.edu, pwensing@nd.edu)

² Wenhao Yu and Tingnan Zhang are with Robotics at Google, Mountain View, CA, USA (magicmelon@google.com, tingnan@google.com)

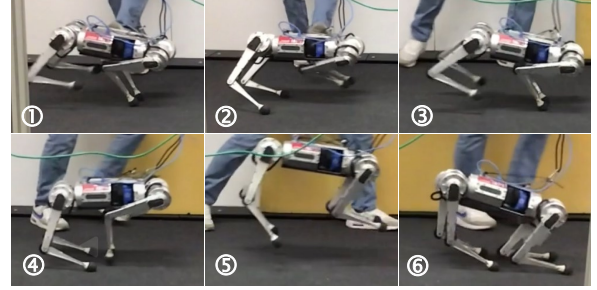


Fig. 1: Time-series snapshots of robust quadruped jumping enabled by the proposed MS-DDP algorithm within an MPC controller. 1. The robot is pushed before jumping. 2. Recovering stability. 3. Back to bounding (1 m/s). 4. Jumping starts. 5. Middle of the jump. 6. Jumping ends.

Differential Dynamic Programming (DDP) [5] is a *single shooting* method to solve the OCP (1), and is shown to have a local quadratic convergence rate [6]. It naturally exploits the temporal structure of the transcribed problem (1) by successively solving a sequence of sub-problems, resulting in linear computational cost with horizon length T . These sub-problems originate from locally solving Bellman’s equation, which additionally gives a local feedback policy without additional computation cost [3]. These properties make DDP exceptionally well suited for MPC with a long prediction horizon. In the past decade, DDP and its variants have been widely used for MPC of quadruped locomotion [3], [7], [8], humanoid balancing [9], [10], and loco-manipulation [11]. The downside of DDP, common with many other single-shooting algorithms, is that an initial guess for the state trajectory cannot be directly supplied, and it is sensitive to the initial guess of the control trajectory [12], [13].

Multiple shooting (Fig. 2) [12], [13] alleviates the sensitivity problem of single shooting by introducing intermediate state variables (also known as *shooting states*). The shooting

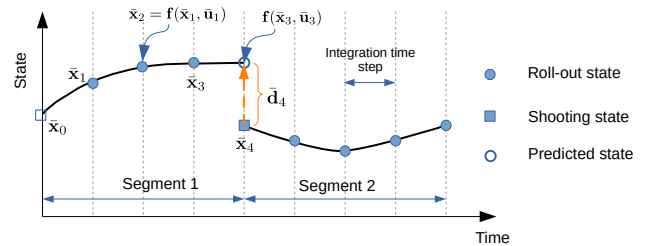


Fig. 2: Illustration of multiple shooting for a nominal trajectory $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$. The shooting state is a decision variable that can be initialized and updated. The roll-out state is always overwritten by dynamics simulation propagated from a previous time instant.

states divide a trajectory into several sub-intervals, known as shooting segments. Numerical integration is performed separately on each segment using the shooting state as an initial condition. Continuity constraints that link the shooting state to a previous segment need to be satisfied at convergence, so that the resulting trajectory is dynamically feasible. The problem sensitivity is distributed across the segments at the price of potential discontinuity (also known as a defect) at each shooting state. This work proposes a new derivation for addressing defects in DDP, which unifies past approaches.

Previous works [14]–[17] on pairing MS and DDP mainly differ in the way the continuity constraints are handled, and can be categorized into two classes. The first class [15], [17] employs Augmented Lagrangian (AL) methods. This approach exhibits low accuracy, but can be addressed using a projection step [15] or a second-order method for the update of the Lagrange multiplier [17]. The second class is inspired by Sequential Quadratic Programming (SQP) [14], [16], which successively linearizes the constraints and solves a quadratic problem. This approach shows good convergence near the (local) solution, but care must be taken when the initial guess is far away from the solution. Globalization strategies such as employing filter-based line search methods [18] or merit functions [19], [20] can address this problem.

A. Contributions

The contributions of this paper are as follows. First, we provide a novel derivation that naturally embeds the multiple-shooting formulation into the DDP algorithm. This derivation enables incorporating second-order sensitivity information across shooting segments, resulting in local quadratic convergence. Further, the derivation enables unifying multiple previous algorithms [14], [16], and opens the door to broadly applicable algorithmic improvements. Second, several algorithmic enhancements are introduced to improve the algorithm's robustness, including a penalty method, a more reliable acceptance condition, and an adaptive merit function. We numerically benchmark the proposed enhancements together with previous algorithms [14], [16] in terms of robustness and convergence rate on several robotic systems. Finally, we implement the algorithm for Model Predictive Control (MPC) of a quadruped robot, and demonstrate its superior performance to single-shooting DDP (SS-DDP).

II. MULTIPLE-SHOOTING FORMULATION OF OPTIMAL CONTROL PROBLEMS

Multiple shooting transcribes the continuous-time OCP (1) to a discrete-time OCP

$$\min_{\mathbf{U}, \mathbf{X}} J(\mathbf{X}, \mathbf{U}) = \sum_{k=0}^{N-1} \ell_k(\mathbf{x}_k, \mathbf{u}_k) + \phi(\mathbf{x}_N) \quad (2a)$$

$$\text{subject to } \underbrace{\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}}_{\mathbf{d}_{k+1}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})} = 0 \quad (2b)$$

where $0 \leq k \leq N$ denotes time instant with $t_N = T$, $\mathbf{U} = \{\mathbf{u}_k\}_{k=0}^{N-1}$, $\mathbf{X} = \{\mathbf{x}_k\}_{k=0}^N$, $\ell_k(\mathbf{x}_k, \mathbf{u}_k)$ the discretized running cost function, $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ the discretized dynamics

function, and $\mathbf{d}_{k+1}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})$ the defect function that measures the dynamics feasibility of a triplet $(\mathbf{x}_k, \mathbf{u}_k, \mathbf{x}_{k+1})$. The formulation (2) considers both the control and state as decision variables. We follow the hybrid scheme in [14] to categorize the state variables to two types, namely a shooting state and a roll-out state. If a state variable \mathbf{x}_{k+1} is a shooting state, then it can be independently initialized and updated, thus the defect function \mathbf{d}_{k+1} is not necessarily zero until the algorithm converges. If a state variable is a roll-out state, then it is always overwritten by the dynamics simulation propagated from a previous time step, thus the defect function is identically equal to zero.

Figure 2 further illustrates these definitions. A trajectory of length $N-1$ is evenly divided into M segments, each incorporating $m = (N-1)/M$ time steps. The initial state of each segment (i.e., at time index $k = m, 2m, \dots, (M-1)m$) comprises the set of shooting states, with time indices denoted \mathbb{I} , whereas the complementary set $\tilde{\mathbb{I}} = \{0, \dots, N\} \setminus \mathbb{I}$ denotes the indices of roll-out states. Given a nominal trajectory $(\bar{\mathbf{X}}, \bar{\mathbf{U}})$, dynamics simulation is performed on each segment using the corresponding shooting state as the initial condition. If $k+1 \in \tilde{\mathbb{I}}$, then $\bar{\mathbf{x}}_{k+1}$ is overwritten by $\mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$. Otherwise, $\bar{\mathbf{x}}_{k+1}$ remains unchanged, and the defect $\mathbf{d}_{k+1} = \mathbf{d}_{k+1}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k, \bar{\mathbf{x}}_{k+1})$ is measured. We denote the nominal value of a variable with $\bar{\cdot}$ overhead throughout the paper. The optimization algorithms developed in this work for problem (2) will need to drive the defects to zero while minimizing the cost function.

III. DIFFERENTIAL DYNAMIC PROGRAMMING FOR MULTIPLE-SHOOTING OCP

DDP applies a local version of Bellman's principle of optimality to the OCP (2) by considering a small perturbation $(\delta\mathbf{X}, \delta\mathbf{U})$ near the nominal trajectory $(\bar{\mathbf{X}}, \bar{\mathbf{U}})$

$$v_k(\delta\mathbf{x}_k) = \min_{\delta\mathbf{u}_k} \underbrace{(\delta\ell_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k) + v_{k+1}(\delta\mathbf{x}_{k+1}))}_{Q_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k)} \quad (3)$$

where $v_k(\cdot)$ denotes the local value function, $\delta\ell_k(\delta\mathbf{x}_k, \delta\mathbf{u}_k)$ quadratically approximates the perturbation of $\ell_k(\mathbf{x}_k, \mathbf{u}_k)$ at $(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)$. We use $Q_k(\cdot, \cdot)$ to denote the Bellman objective of eq. (3) for simplicity. In the traditional case of SS-DDP, the algorithm considers every state as a roll-out state, thus the perturbation $(\delta\mathbf{X}, \delta\mathbf{U})$ needs to satisfy

$$\delta\mathbf{x}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k + \delta\mathbf{x}_k, \bar{\mathbf{u}}_k + \delta\mathbf{u}_k) - \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k). \quad (4)$$

Substituting eq. (4) to the local Bellman's equation (3) and approximating \mathbf{f} quadratically results in standard Ricatti-like difference equations and a local policy [5], [21].

A. Backward Sweep Accounting For Defect

Traditional DDP, however, is not applicable to deal with the shooting state as introduced in the formulation (2) because of the defect. In this work, we revise the traditional DDP backward sweep to account for the defect using a simple but effective trick. Eq. (4) is modified as below

$$\delta\mathbf{x}_{k+1} = \mathbf{f}(\bar{\mathbf{x}}_k + \delta\mathbf{x}_k, \bar{\mathbf{u}}_k + \delta\mathbf{u}_k) - \bar{\mathbf{x}}_{k+1}. \quad (5)$$

The equation (5) allows for dynamic infeasibility in the initial guess or intermediate trajectories $(\bar{\mathbf{X}}, \bar{\mathbf{U}})$. To derive the update policy associated with (5), we approximate $v_k(\delta \mathbf{x}_k)$ to the second order as in DDP

$$v_k(\delta \mathbf{x}_k) = \frac{1}{2}(\delta \mathbf{x}_k)^\top \mathbf{S}_k \delta \mathbf{x}_k + \mathbf{s}_k^\top \delta \mathbf{x}_k + s_k. \quad (6)$$

where \mathbf{S}_k , \mathbf{s}_k , s_k are the Hessian, gradient, and zero-order terms of $v_k(\delta \mathbf{x}_k)$. By approximating \mathbf{f} in eq. (5) to the second order and substituting (6) into eq. (3), we get

$$Q_k(\delta \mathbf{x}, \delta \mathbf{u}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}^\top \begin{bmatrix} 0 & \mathbf{Q}_{\mathbf{x},k}^\top & \mathbf{Q}_{\mathbf{u},k}^\top \\ \mathbf{Q}_{\mathbf{x},k} & \mathbf{Q}_{\mathbf{xx},k} & \mathbf{Q}_{\mathbf{ux},k}^\top \\ \mathbf{Q}_{\mathbf{u},k} & \mathbf{Q}_{\mathbf{ux},k} & \mathbf{Q}_{\mathbf{uu},k} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}, \quad (7)$$

where

$$\mathbf{Q}_{\mathbf{x},k} = \mathbf{q}_k + \mathbf{A}_k^\top (\mathbf{s}_{k+1} + \mathbf{S}_{k+1} \bar{\mathbf{d}}_{k+1}), \quad (8a)$$

$$\mathbf{Q}_{\mathbf{u},k} = \mathbf{r}_k + \mathbf{B}_k^\top (\mathbf{s}_{k+1} + \mathbf{S}_{k+1} \bar{\mathbf{d}}_{k+1}), \quad (8b)$$

$$\mathbf{Q}_{\mathbf{xx},k} = \mathbf{Q}_k + \mathbf{A}_k^\top \mathbf{S}_{k+1} \mathbf{A}_k + \mathbf{S}_{k+1} \cdot \mathbf{f}_{\mathbf{xx},k}, \quad (8c)$$

$$\mathbf{Q}_{\mathbf{uu},k} = \mathbf{R}_k + \mathbf{B}_k^\top \mathbf{S}_{k+1} \mathbf{B}_k + \mathbf{S}_{k+1} \cdot \mathbf{f}_{\mathbf{uu},k}, \quad (8d)$$

$$\mathbf{Q}_{\mathbf{ux},k} = \mathbf{P}_k + \mathbf{B}_k^\top \mathbf{S}_{k+1} \mathbf{A}_k + \mathbf{S}_{k+1} \cdot \mathbf{f}_{\mathbf{ux},k}, \quad (8e)$$

in which $\mathbf{A}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}|_{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}$, $\mathbf{B}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}|_{(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)}$, \mathbf{q}_k and \mathbf{r}_k are gradients of ℓ_k w.r.t. \mathbf{x} and \mathbf{u} respectively, \mathbf{Q}_k , \mathbf{R}_k , and \mathbf{P}_k are second-order partials of ℓ_k , and $\mathbf{f}_{(\cdot,\cdot)}$ are second-order partials of \mathbf{f} . The recursive equations for \mathbf{S}_k , \mathbf{s}_k , and s_k are

$$\mathbf{S}_k = \mathbf{Q}_{\mathbf{xx},k} - \mathbf{Q}_{\mathbf{ux},k}^\top \mathbf{Q}_{\mathbf{uu},k}^{-1} \mathbf{Q}_{\mathbf{ux},k} \quad (9a)$$

$$\mathbf{s}_k = \mathbf{Q}_{\mathbf{x},k} - \mathbf{Q}_{\mathbf{ux},k}^\top \mathbf{Q}_{\mathbf{uu},k}^{-1} \mathbf{Q}_{\mathbf{u},k}, \quad (9b)$$

$$s_k = s_{k+1} - \frac{1}{2} \mathbf{Q}_{\mathbf{u},k}^\top \mathbf{Q}_{\mathbf{uu},k}^{-1} \mathbf{Q}_{\mathbf{u},k} + \mathbf{s}_{k+1}^\top \bar{\mathbf{d}}_k + \frac{1}{2} \bar{\mathbf{d}}_{k+1}^\top \mathbf{S}_{k+1} \bar{\mathbf{d}}_{k+1}, \quad (9c)$$

with the boundary conditions $\mathbf{S}_N = \mathbf{Q}_N$, $\mathbf{s}_N = \mathbf{q}_N$, $s_N = 0$. Minimizing (7) over $\delta \mathbf{u}_k$ results in a local optimal control policy

$$\delta \mathbf{u}_k^* = \delta \tilde{\mathbf{u}}_k + \mathbf{K}_k \delta \mathbf{x}_k, \quad (10)$$

where

$$\delta \tilde{\mathbf{u}}_k = -\mathbf{Q}_{\mathbf{uu},k}^{-1} \mathbf{Q}_{\mathbf{u},k}, \quad \mathbf{K}_k = -\mathbf{Q}_{\mathbf{uu},k}^{-1} \mathbf{Q}_{\mathbf{ux},k}. \quad (11)$$

The equations (8) and (9) provide general formulas for the backward sweeps of four DDP variants

- 1) *MS-DDP* (this work): Multiple shooting, second-order.
- 2) *SS-DDP* [5], [22]: Single shooting, second-order (no blue).
- 3) *MS-iLQR* [14], [16]: Multiple shooting, first-order (no red).
- 4) *SS-iLQR* [21], [23]: Single shooting, first-order (no blue and no red).

We will investigate the effect of the second-order dynamics on local convergence in Section VI-A.

B. Forward Roll-out

The control policy (11) provides a search direction for the control update from iteration to iteration. A forward roll-out for the dynamics must be conducted to obtain a search direction for the state update. In this section, we explore three methods for the forward roll-out that mainly differ in the dynamics, namely a *linear roll-out* [14], [24], *nonlinear roll-out* [16], and *hybrid roll-out* [14]. For clarity, we repeat some notations here, $(\bar{\mathbf{X}}, \bar{\mathbf{U}})$ denotes the nominal trajectory, and $(\mathbf{X}', \mathbf{U}')$ denotes the new trajectory. Whichever roll-out method is used, the control update always has the same format

$$\mathbf{u}'_k = \bar{\mathbf{u}}_k + \underbrace{\alpha \delta \tilde{\mathbf{u}}_k + \mathbf{K}_k (\mathbf{x}'_k - \bar{\mathbf{x}}_k)}_{\delta \mathbf{u}_k(\alpha)}. \quad (12)$$

where $\mathbf{x}'_0 = \bar{\mathbf{x}}_0$, $\alpha \in (0, 1]$ is the step size, which is used in backtracking line search for global convergence [21], and $\delta \mathbf{u}_k(\alpha)$ is the scaled search direction. The main difference between the three roll-out methods is in the state update.

A *linear roll-out* simulates the linearized dynamics of (5) using the control policy (11)

$$\mathbf{x}'_{k+1} = \bar{\mathbf{x}}_{k+1} + \underbrace{[\mathbf{A}_k (\mathbf{x}'_k - \bar{\mathbf{x}}_k) + \mathbf{B}_k \delta \mathbf{u}_k(\alpha)] + \alpha \bar{\mathbf{d}}_{k+1}}_{\delta \mathbf{x}_{k+1}(\alpha)} \quad (13)$$

where $\delta \mathbf{x}_{k+1}(\alpha)$ is the scaled search direction for $\bar{\mathbf{x}}_{k+1}$. Note that $\delta \mathbf{x}_{k+1}(\alpha)$ scales linearly with α in this case. This method is computationally cheap, since it only needs to be executed once with $\alpha = 1$, and a line search can then be performed in parallel across all time instants. The downside, however, is that the method simplifies the nonlinear dynamics, and thus is subject to prediction error [16]. Further, the linear roll-out requires that every state is a *shooting state*, losing the flexibility for other algorithmic options.

To account for nonlinearity, a *nonlinear roll-out* [16] simulates the original nonlinear dynamics using the control policy (11)

$$\mathbf{x}'_{k+1} = \bar{\mathbf{x}}_{k+1} + \underbrace{[\mathbf{f}(\mathbf{x}'_k, \mathbf{u}'_k) - \mathbf{f}(\bar{\mathbf{x}}_k, \bar{\mathbf{u}}_k)] + \alpha \bar{\mathbf{d}}_{k+1}}_{\delta \mathbf{x}_{k+1}(\alpha)} \quad (14)$$

The nonlinear roll-out (14) avoids the linear prediction error, with the above equivalent to the scheme in [16]. The derivation here, however, more resembles the behavior of (13) with a replacement of nonlinear dynamics. Unlike the linear roll-out, which enables parallel computation, the nonlinear roll-out has to be performed serially, thus potentially hindering the computational performance.

A *hybrid roll-out* method was proposed in [14], which attempts to combine the benefit of both. Namely, the hybrid method first performs a linear roll-out (13) to obtain $\delta \mathbf{x}_{k+1}(\alpha)$ for the shooting states. The updated shooting state is then used as an initial condition, and the nonlinear roll-out can be performed independently on each shooting segment for the roll-out states. With this method, the search direction for the shooting nodes only needs to be computed once, and the line search can then be performed in parallel on each shooting segment. Previous work [14] has shown that the

hybrid roll-out exhibits better global convergence than the linear roll-out, thus the linear roll-out is not considered here.

The unified MS-DDP framework developed in this work synthesizes all four DDP variants for the backward pass, and the nonlinear and hybrid forward roll-out. The multiple algorithm configurations enabled by this synthesis may produce different convergence behaviors as will be shown in Section VI-A.

IV. IMPROVING ROBUSTNESS AND FLEXIBILITY

The previous section introduced methods for computing the search direction that updates the control and state variables. This section presents new techniques that better measure the quality of a search step to determine the step size. Further, we introduce an advance that alters the search direction by modifying the backward sweep (eq. (9)).

A. Merit Function

MS-DDP needs to balance two goals, minimizing the cost function (2a) while decreasing the defects (2b). These two goals can sometimes be conflicting with each other since reducing the defect may otherwise increase the cost [25]. A merit function synthesizes the two objectives into one single function, and is widely used in Nonlinear Programming (NLP) for constrained optimization [25], [26]. In this work, we use a merit function to monitor the progress of MS-DDP. The merit function considered is an L_p -norm merit function since it has been proven to be exact [25] and does not require an estimate of the Lagrange multiplier. The L_p -norm merit function for problem (2) is defined as

$$M(\mathbf{X}, \mathbf{U}) = J(\mathbf{X}, \mathbf{U}) + \mu \|\mathbf{d}(\mathbf{X}, \mathbf{U})\|_p \quad (15)$$

where $\mathbf{d}(\cdot, \cdot)$ is a vector aggregating all the defects, $\mu > 0$ is a weighting parameter that balances the cost function and the defects violation. To avoid meticulous tuning of μ for different problems, we consider an adaptive scheme that is motivated by [25] so that μ is updated as

$$\mu = \frac{EC(\alpha)}{(1 - \rho) \|\mathbf{d}(\bar{\mathbf{X}}, \bar{\mathbf{U}})\|_p} + \mu_0 \quad \text{when} \quad \|\mathbf{d}(\bar{\mathbf{X}}, \bar{\mathbf{U}})\|_p > \kappa_d \quad (16)$$

where $\mu_0 > 0$ sets a safety margin, $EC(\alpha)$ denotes the expected cost change due to $(\delta\mathbf{X}(\alpha), \delta\mathbf{U}(\alpha))$, which is discussed in the next subsection, $0 < \rho < 1$ is a fixed tuning parameter, and $\kappa_d > 0$ is the threshold for updating μ . We use $\rho = 0.5$, and $\mu_0 = 10$ for all problems in this work without further tuning. An alternative approach to using a merit function is a filter-based technique [18], [27].

B. Acceptance Condition

A simple condition to accept a search step $(\delta\mathbf{X}(\alpha), \delta\mathbf{U}(\alpha))$ is to ensure the merit function (15) is decreased, i.e., $M(\mathbf{X}', \mathbf{U}') - M(\bar{\mathbf{X}}, \bar{\mathbf{U}}) < 0$. This requirement, however, may not produce convergence to a local optimum, as shown in [25]. In this work, we use an Armijo condition to impose sufficient merit reduction

$$M(\mathbf{X}', \mathbf{U}') < M(\bar{\mathbf{X}}, \bar{\mathbf{U}}) + \gamma (EC(\alpha) - \alpha \mu \|\mathbf{d}(\bar{\mathbf{X}}, \bar{\mathbf{U}})\|_p) \quad (17)$$

where $0 < \gamma < 1$ is a tuning parameter, $EC(\alpha)$ is the expected cost change, and

$$EC(\alpha) = \alpha EC_1 + \frac{1}{2} \alpha^2 EC_2 \quad (18)$$

in which

$$EC_1 = \mathbf{q}_N^\top \delta \mathbf{x}_N^l + \sum_{k=0}^{N-1} \mathbf{q}_k^\top \delta \mathbf{x}_k^l + \mathbf{r}_k^\top \delta \mathbf{u}_k^l, \quad (19a)$$

$$EC_2 = \delta \mathbf{x}_N^{l\top} \mathbf{Q}_N \delta \mathbf{x}_N^l + \sum_{k=0}^{N-1} \delta \mathbf{x}_k^{l\top} \mathbf{Q}_k \delta \mathbf{x}_k^l + \delta \mathbf{u}_k^{l\top} \mathbf{R}_k \delta \mathbf{u}_k^l + \delta \mathbf{u}_k^{l\top} \mathbf{P}_k \delta \mathbf{x}_k^l \quad (19b)$$

where $\delta \mathbf{u}_k^l$ and $\delta \mathbf{x}_k^l$ are obtained via the linear roll-out (12) and (13). The EC (18) provides an exact cost change in the case of linear dynamics and quadratic cost approximation. It is equivalent to the model of expected cost change [21] computed from the iLQR backward pass if all defects are zero. We find that this expectation model further improves upon the one used within [16], [28], in the sense that cost effects from the defect are fully treated, leading to an exact match when applied to linear quadratic MS problems.

C. Modified Backward Sweep Using A Penalty Method

A backtracking line search is employed to determine the step size α . The step size α , however, may still be so small that the global convergence is hindered. In this section, we introduce a penalty method to improve this process. Note that in Fig. 2 the shooting state adjusts the defect size only from the right side, but the rolled-out state $\mathbf{x}_{k+1}^- := \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ on the left side is not aware of this change. The proposed method strategically controls the defect from both sides, by imposing a penalty that promotes connecting the segments from the left side as well. This is done by adding the penalty term $\|\mathbf{x}_{k+1}^- - \bar{\mathbf{x}}_{k+1}\|_{\mathbf{Q}_{d_{k+1}}}^2$ to the cost function in (2), where $\mathbf{Q}_{d_{k+1}}$ is a positive definite weight matrix if $k+1$ is a shooting state, and is zero if $k+1$ is a roll-out state. Adding the penalty term to (2) amounts to modifying the backward sweep equations (8b) and (8a) such that

$$\mathbf{s}_{k+1} \leftarrow \mathbf{s}_{k+1} - \mathbf{Q}_d \bar{\mathbf{d}}_k \quad (20a)$$

$$\mathbf{S}_{k+1} \leftarrow \mathbf{S}_{k+1} + \mathbf{Q}_d \quad (20b)$$

This penalty method is similar in spirit to the proximal term [29] that helps improve the conditioning of the KKT system. The benefit of adding the terminal cost to the algorithm performance will be demonstrated in the result section.

V. DISCUSSION ON THE UNIFIED PERSPECTIVE

The search direction computation in Section III and the globalization method for step acceptance in Section IV comprise one iteration of the MS-DDP algorithm. A brief summary of the overall MS-DDP framework is given below

- 1) Give the nominal trajectory $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$, optimization horizon N , and number of shooting segments M .
- 2) Perform dynamics approximation and cost function approximation around the nominal trajectory.

TABLE I: Summary of algorithmic configurations with the unified MS-DDP

	Unified MS-DDP	FiLQR [16]	GN-iLQR [14]
Backward Sweep Dyn (8)	SS, MS 1 st order 2 nd order	SS, MS 1 st order	SS, MS 1 st order
Forward Roll-out	Hybrid Nonlinear	Nonlinear	Hybrid
Merit Function	Adaptive Constant Cost function	Cost function	Constant
Expectaion Model	Exact Approximate	Approximate	Simple

- 3) Compute the optimal control policy (11) using one of the four DDP variants.
- 4) Perform backtracking line search using either hybrid roll-out or nonlinear roll-out, the adaptive merit function (16) and the acceptance condition (17).
- 5) Check the cost convergence criterion and feasibility. Go to 2) if not satisfied, and terminate otherwise.

The MS-DDP framework herein provides a unified perspective since it enables multiple configurations, as summarized in Table I. Four DDP variants could be used in performing the backward sweep, and two forward roll-out methods could be employed for line search. Further, the penalty method and the adaptive merit function offer additional options for improving the algorithm robustness and flexibility. As we will show in the next section, the performance of an algorithm to solve (2) is problem-dependent. The unified MS-DDP framework enables easy comparison across different algorithm configurations. In fact, if we choose the MS-iLQR for computing the backward sweep (eq. (8) and eq. (9)), the hybrid roll-out for line search, and $\gamma = 0$ for the acceptance condition, then the proposed framework can be simplified to GN-iLQR [14]. If we choose the MS-iLQR for the backward sweep, the nonlinear roll-out for line search, and compute EC with an approximate model, the MS-DDP framework is then simplified to FiLQR [16]¹.

VI. NUMERICAL RESULTS

The MS-DDP framework is benchmarked on three problems for numerical analysis. Each problem is associated with moving a robotic system from an initial state to a terminal state. A semi-implicit Euler method is used for integration, with the integration time step 0.02 s. Quadratic cost functions are used for all problems. We use the L_2 -norm to measure the total defect. Algorithm convergence is approximately achieved if the normalized cost change is within $1e-8$, and the total defect is less than $1e-3$. We briefly describe each problem here.

- 1) *Acrobot*: A two-link manipulator where only the second joint is actuated. The acrobot needs to swing from a downward configuration up to an upward configuration in four seconds.

¹For consistency with the nomenclature in this paper, we depart from the name “FDDP” used in [16] and use FiLQR instead, which reflects the exclusive use of first-order dynamics sensitivities in the method.

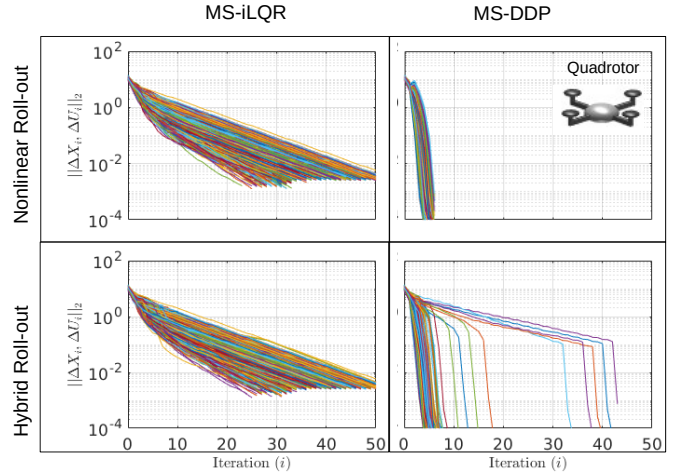


Fig. 3: Local convergence for 1000 randomly sampled initial trajectories of the quadrotor in the neighborhood of the optimal trajectory. Left: MS-iLQR (Gauss-Newton Hessian approximation). Right: MS-DDP with full Hessian.

- 2) *Quadrotor*: The quadrotor is modeled as a rigid body with four thrust inputs, each at a certain distance from the center of mass [30]. The quadrotor is supposed to travel 5 m from one static position to another static position in four seconds.
- 3) *Manipulator*: The Kuka iiwa 7-DoF serial manipulator is used. The robot needs to swing from an upward configuration to a bending configuration in four seconds.

A. Numerical Convergence Analysis

The local convergence rate of an algorithm is defined in the neighborhood of a local optimum, whereas global convergence is characterized by its capability to move a remote initial guess to a local optimum. In this section, we statistically quantify these properties for the MS-DDP framework configured with and without second-order dynamics.

1) *Local Convergence*: Denote $(\mathbf{X}^*, \mathbf{U}^*)$ a local optimum of problem (2), $(\mathbf{X}_j, \mathbf{U}_j)$ the j^{th} iterate produced by the MS-DDP framework, and $(\Delta\mathbf{X}_j, \Delta\mathbf{U}_j)$ the difference between the j^{th} iterate and the local optima $(\mathbf{X}^*, \mathbf{U}^*)$. The rate of local convergence is characterized by

$$\lim_{j \rightarrow \infty} \|(\Delta\mathbf{X}_{j+1}, \Delta\mathbf{U}_{j+1})\|_2 = \kappa \|(\Delta\mathbf{X}_j, \Delta\mathbf{U}_j)\|_2^\epsilon \quad (21)$$

where $\epsilon = 1$ and $0 < \kappa < 1$ indicates linear convergence, and $\epsilon = 2$ and $\kappa > 0$ indicates quadratic convergence. Four MS-DDP configurations are studied here: with and without the second-order dynamics in eq. (8), and hybrid vs. nonlinear roll-out. All algorithms are benchmarked on the quadrotor problem. Each algorithm is configured to have 200 shooting segments, and is tested in a Monte Carlo fashion with 1000 initial guesses. The state guess is randomly sampled from a uniform distribution around \mathbf{X}^* while the control guess remains the same as \mathbf{U}^* . Figure 3 depicts the local convergence results for all four algorithms. Linear convergence is always obtained if only first-order dynamics is used in (eq. (8)). Adding the second-order dynamics can improve the convergence rate. Quadratic convergence is consistently

achieved over all samples with the nonlinear roll-out. For MS-DDP with the hybrid roll-out, the local convergence rate for certain samples is somewhere between linear convergence and quadratic convergence. This difference is reasonable in the sense that the shooting state update is based on the linearized dynamics with the hybrid roll-out, thus subject to prediction error, whereas it is based on nonlinear dynamics with the nonlinear roll-out, thus free of prediction error.

B. Effects of Acceptance Condition

We show the effect of the acceptance condition on the algorithm convergence using the acrobot problem. The proposed exact EC (18) is compared against the approximated EC as used by FiLQR [16]. Three algorithms are evaluated, FiLQR, FiLQR-exact, and MS-iLQR. FiLQR-exact differs from FiLQR only in the expected cost change. MS-iLQR is configured to use the adaptive merit function and the exact EC (18), and shares everything else (backward sweep and nonlinear roll-out) in common with FiLQR. All three algorithms are configured to regard all state variables as shooting states, i.e., $M = N - 1$. The initial state trajectories are obtained by linearly interpolating the initial and terminal states, while the initial controls are zero.

Figure. 4 shows the comparison results in terms of cost convergence and dynamics feasibility. FiLQR fails to make progress on the given initial trajectory, whereas both the FiLQR-exact and MS-iLQR converge in 25 iterations, demonstrating the benefit of using the exact EC (18). To have a better understanding, the actual change in cost and the expected cost change are compared for FiLQR and FiLQR-exact. To do so, we run FiLQR-exact for several iterations, and perform a line search for both methods. The results are shown in Fig. 5. For FiLQR, the difference between the actual change and expected change is obvious, whereas for

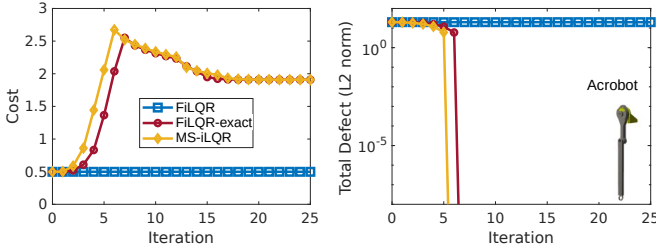


Fig. 4: Effects of expected cost change on the global convergence for the acrobot with different algorithm configurations: FiLQR, FiLQR with exact expected cost change (eq. (18)), MS-iLQR with adaptive merit function and nonlinear roll-out.

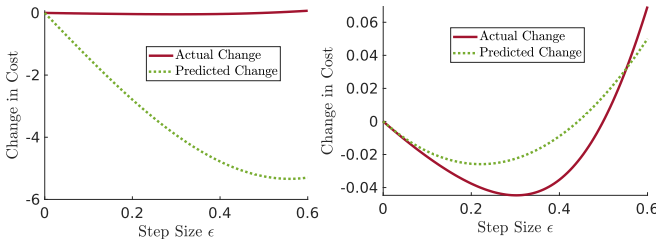


Fig. 5: Expected cost change and actual cost change for the acrobot vs. step size. Left: FiLQR. Right: FiLQR with the proposed exact EC (eq. (18)).

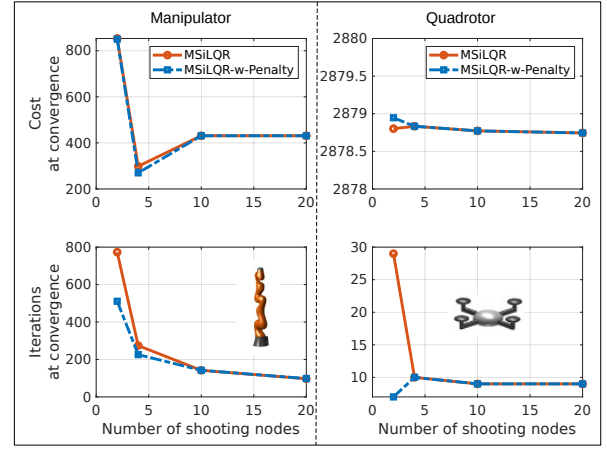


Fig. 6: Effects of penalty method on algorithm convergence with a varying number of shooting nodes, benchmarked on the manipulator and quadrotor examples. Benefits are observed only for a low number of shooting nodes.

FiLQR-exact, the actual change and expected change share the same slope and concavity at step size 0, demonstrating the proposed EC is more accurate. The MS-DDP framework thus is configured to use (18) for the rest of this work.

C. Effects of Penalty Method

Roughly speaking, for the multiple-shooting OCP (2), the fewer the number of shooting segments, the more nonlinear the problem (2) is. This subsection investigates the effect of the penalty method on algorithm convergence rate for problems with different nonlinearity. To do so, we employ a varying number of shooting segments for the manipulator and the quadrotor examples. Figure 6 illustrates the results in terms of the cost at convergence and number of iterations to converge, acquired with and without the penalty method. MS-iLQR is configured with a hybrid roll-out and the adaptive merit function. For each example, similar costs are achieved at convergence with and without the penalty method, given the same number of shooting nodes. The penalty method, however, significantly reduces the number of iterations for the case of two shooting nodes, where higher nonlinearity arises compared to the case of more shooting nodes. Though the penalty method does not show obvious performance improvement for the quadrotor problem with four shooting nodes and above, it enables the more nonlinear manipulator problem to achieve 48 fewer iterations to converge with four shooting nodes. These observations indicate that the proposed penalty method is helpful to promote faster convergence for more nonlinear problems. Closer examinations reveal that larger step sizes are enabled with the penalty method. This result is not surprising, since both sides of the defect are considered in computing the search direction, thus facilitating faster convergence.

VII. MS-DDP FOR MPC ON A QUADRUPEL ROBOT

One motivation for this work is to develop a robust and efficient solver for real-time MPC. This section investigates the performance of the MS-DDP framework in the context of MPC for dynamic quadruped locomotion. The MPC problem

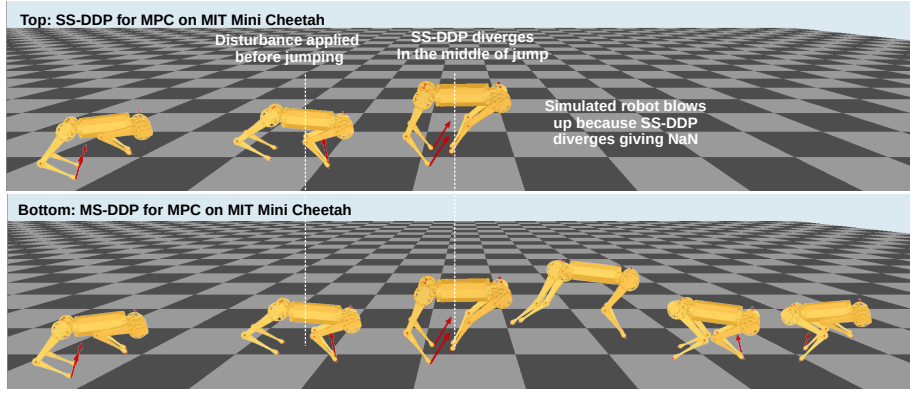


Fig. 7: Snapshots of the running-to-jump gait captured in the Mini Cheetah simulator. Kick disturbance to the lateral velocity is applied before jumping. The MS-DDP (bottom) succeeds in recovering the robot from the kick, whereas the SS-DDP (top) diverges in the middle of the jump.

is constructed based upon a hybrid kinodynamics (HKD) model [7], which reasons about then trunk dynamics and contact-dependent leg kinematics. In previous work [7], the HKD-MPC problem was solved using a SS-DDP variant that is tailored for hybrid systems [31]. In this work, we adapt the previous solver to use MS-DDP, and compare its performance against the previous SS-DDP implementation.

A. Simulation Results

The comparisons are conducted on the MIT Mini Cheetah [32] in a high-fidelity simulator for multiple gaits, including mildly dynamic gaits (e.g., trotting and bounding), and highly dynamic motions (e.g., jumping). In all cases, the HKD-MPC (50 Hz) runs asynchronously from the low-level controller (500 Hz). The prediction horizon was chosen as 0.5 s with an integration time step of 10 ms. At each MPC control step, the DDP solver is terminated after two iterations.

SS-DDP and MS-DDP perform equivalently well for trotting and bounding, which is reasonable since the feedback policy may be sufficient to prevent each solver from diverging for mildly-dynamic gaits. The same equivalence was not observed for the more dynamic behavior of jumping. For this motion, the robot accelerates to 2 m/s using a bounding gait, makes a jump at 2.5 s with a duration of 0.35 s, and recovers to bounding. A 0.4 m/s lateral velocity disturbance is injected before the jump. The MS-DDP enables the robot to recover stability, whereas the SS-DDP does not due to divergence in the middle of the jump. The snapshots of this result are shown in Fig. 7. To understand this difference, the accumulated costs (i.e., evaluation of the cost function (2a)) are shown in Fig. 8. Both tend to increase during the jump, but SS-DDP quickly diverges. Though MS-DDP encounters a relatively-large dynamics infeasibility (i.e., total defect of 3) during the jump, this violation is temporary and is helpful to keep the accumulated cost bounded. To unveil the cause of SS-DDP divergence, we check the nominal roll-out trajectory of each algorithm at the MPC step immediately before SS-DDP diverges. Note that the roll-out trajectory is along the prediction horizon, as opposed to along the MPC step in Fig. 8. The roll rate and the roll angle are depicted in Fig. 9. At about 0.36 s, a foot contact is established, and the nominal

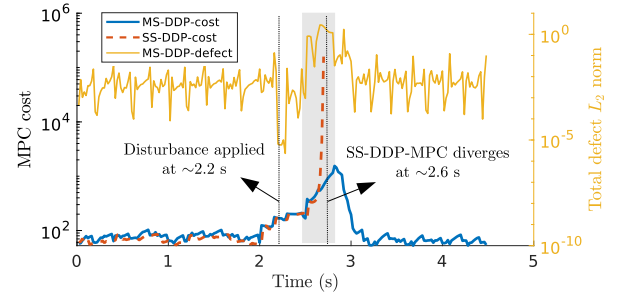


Fig. 8: Accumulated cost of SS-DDP and MS-DDP, and total defect of MS-DDP at each MPC step. The defect of SS-DDP is always zero since it is inherently dynamically feasible. The grey area indicates the jumping period.

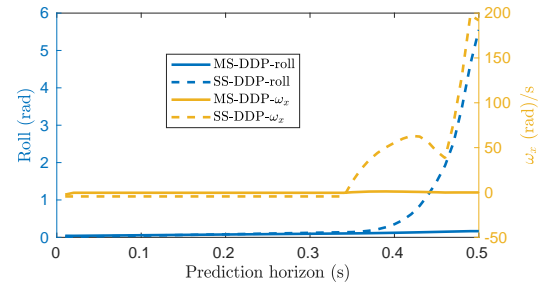


Fig. 9: Nominal roll-out of SS-DDP and MS-DDP along the prediction horizon at the MPC step immediately before SS-DDP diverges. The roll angle and the angular velocity around x -axis are depicted.

control policy (with feedback) of SS-DDP fails to stabilize the roll motion, thus the initial state trajectory diverges. By contrast, the MS-DDP does not suffer from this problem since its state trajectory is warm-started as well.

The results shown in Figs. 7, 8 and 9 demonstrate the superior performance of MS-DDP over SS-DDP for highly dynamic locomotion. This conclusion is aligned with [18], which focuses on enabling the robot to traverse more complex environments using multiple shooting.

B. Hardware Results

We qualitatively validate the performance of MS-DDP on the Mini Cheetah hardware. The control setup in hardware largely matches the simulation setup with the following exceptions. First, the HKD-MPC is executed at 100 Hz

on hardware. We found that 50 Hz was not sufficient to stabilize the jumping motion largely due to the increased model mismatch of the hardware. An alternative approach could be to employ a QP-based whole-body controller at a high rate. To avoid crossing singularity for the swing legs, we reduce the desired forward velocity to 1.0 m/s. Further, we manually push the robot to imitate the velocity disturbance. Time-series snapshots of hardware results are shown in Fig. 1, while the complete results are in the accompanying video.

VIII. CONCLUSIONS

This work presents a unified framework for extending DDP to a multiple-shooting OCP solver. The proposed framework provides multiple configurations and several enhancements, allowing for easy comparison with and between previous algorithms. The novel derivation of the defect-aware DDP backward pass enables using second-order dynamics, and is shown to have local quadratic convergence when used with the nonlinear roll-out method. We show that the expected cost change model is important for algorithm convergence, and propose an exact model that further improves the performance of a state-of-the-art solver. A penalty method is introduced to provide additional robustness for problems with higher nonlinearity, and is shown to be effective in the case of a small number of shooting nodes. Future work will focus on generalizing the results to more problems with broader statistical assessment.

IX. ACKNOWLEDGEMENTS

The Mini Cheetah is sponsored by the MIT Biomimetic Robotics Lab and NAVER LABS.

REFERENCES

- [1] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “Pampc: Perception-aware model predictive control for quadrotors,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 1–8.
- [2] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, “Dynamic locomotion in the mit cheetah 3 through convex model-predictive control,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 1–9.
- [3] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, “Feedback mpc for torque-controlled legged robots,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 4730–4737.
- [4] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Penter, T. Koolen, P. Marion, and R. Tedrake, “Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot,” *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.
- [5] D. Mayne, “A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems,” *Int. Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.
- [6] L.-z. Liao and C. A. Shoemaker, “Advantages of differential dynamic programming over newton’s method for discrete-time optimal control problems,” Cornell University, Tech. Rep., 1992.
- [7] H. Li, T. Zhang, W. Yu, and P. M. Wensing, “Versatile real-time motion synthesis via kino-dynamic mpc with hybrid-systems ddp,” *arXiv preprint arXiv:2209.14138*, 2022.
- [8] C. Mastalli, S. Prasad Chhatoi, T. Corbères, S. Tonneau, and S. Vijayakumar, “Inverse-dynamics mpc via nullspace resolution,” *arXiv e-prints*, pp. arXiv–2209, 2022.
- [9] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, “Whole-body model-predictive control applied to the HRP-2 humanoid,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 3346–3351.
- [10] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taïx, and N. Mansard, “Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot,” in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2022, pp. 638–644.
- [11] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, “Whole-body mpc and online gait sequence generation for wheeled-legged robots,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2021, pp. 8388–8395.
- [12] 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.
- [13] M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber, “Fast direct multiple shooting algorithms for optimal robot control,” in *Fast motions in biomechanics and robotics*, 2006, pp. 65–93.
- [14] M. Gifftthaler, M. Neunert, M. Stäubli, J. Buchli, and M. Diehl, “A family of iterative gauss-newton shooting methods for nonlinear optimal control,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 1–9.
- [15] T. A. Howell, B. E. Jackson, and Z. Manchester, “ALTRO: a fast solver for constrained trajectory optimization,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 7674–7679.
- [16] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, “Crocodyl: An efficient and versatile framework for multi-contact optimal control,” in *IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 2536–2542.
- [17] E. Pellegrini and R. P. Russell, “A multiple-shooting differential dynamic programming algorithm. part 1: Theory,” *Acta Astronautica*, vol. 170, pp. 686–700, 2020.
- [18] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, “Perceptive locomotion through nonlinear model predictive control,” *arXiv preprint arXiv:2208.08373*, 2022.
- [19] W. Jallet, A. Bambade, N. Mansard, and J. Carpentier, “Constrained differential dynamic programming: A primal-dual augmented lagrangian approach,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2022, pp. 13 371–13 378.
- [20] S. Singh, J.-J. Slotine, and V. Sindhwani, “Optimizing trajectories with closed-loop dynamic sqp,” in *Int. Conf. on Robotics and Automation*, 2022, pp. 5249–5254.
- [21] Y. Tassa, T. Erez, and E. Todorov, “Synthesis and stabilization of complex behaviors through online trajectory optimization,” in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 4906–4913.
- [22] J. N. Nganga and P. M. Wensing, “Accelerating second-order differential dynamic programming for rigid-body systems,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7659–7666, 2021.
- [23] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, “An efficient optimal planning and control framework for quadrupedal locomotion,” in *IEEE Int. Conf. on Robotics and Automation*, 2017, pp. 93–100.
- [24] W. Jallet, N. Mansard, and J. Carpentier, “Implicit differential dynamic programming,” in *IEEE Int. Conf. on Robotics and Automation*, 2022, pp. 1455–1461.
- [25] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.
- [26] P. E. Gill, W. Murray, and M. A. Saunders, “SNOPT: An sqp algorithm for large-scale constrained optimization,” *SIAM review*, vol. 47, no. 1, pp. 99–131, 2005.
- [27] A. Pavlov, I. Shames, and C. Manzie, “Interior point differential dynamic programming,” *IEEE Transactions on Control Systems Technology*, vol. 29, no. 6, pp. 2720–2727, 2021.
- [28] N. Mansard, “Feasibility-prone differential dynamic programming – is DDP a multiple shooting algorithm?” <https://github.com/loco-3d/crocodyl/blob/master/doc/reports/fddp/root.tex>, 2020.
- [29] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, “Prox-qp: Yet another quadratic programming solver for robotics and beyond,” in *RSS 2022-Robotics: Science and Systems*, 2022.
- [30] H. Li, R. J. Frei, and P. M. Wensing, “Model hierarchy predictive control of robotic systems,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3373–3380, 2021.
- [31] H. Li and P. M. Wensing, “Hybrid systems differential dynamic programming for whole-body motion planning of legged robots,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448 – 5455, 2020.
- [32] B. Katz, J. Di Carlo, and S. Kim, “Mini cheetah: A platform for pushing the limits of dynamic quadruped control,” in *Int. Conf. on Robotics and Automation*, 2019, pp. 6295–6301.