DOC: Differentiable Optimal Control for Retargeting Motions onto Legged Robots

RUBEN GRANDIA*, Disney Research, Switzerland
FARBOD FARSHIDIAN*, ETH Zürich, Switzerland
ESPEN KNOOP, Disney Research, Switzerland
CHRISTIAN SCHUMACHER, Disney Research, Switzerland
MARCO HUTTER, ETH Zürich, Switzerland
MORITZ BÄCHER, Disney Research, Switzerland

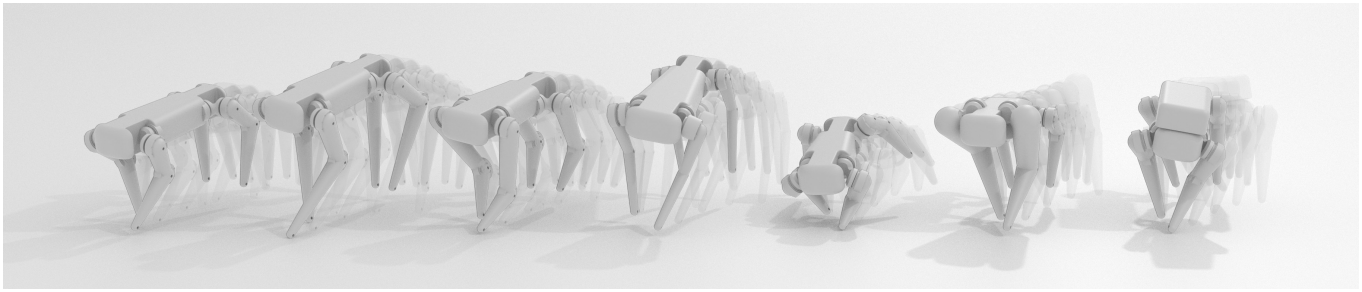

Fig. 1. Our differentiable optimal control enables the retargeting of expressive motions, taken from either animals or animations, onto legged robots of vastly different proportions and mass distribution, as illustrated here with a dog motion retargeted onto seven different quadrupeds.

Legged robots are designed to perform highly dynamic motions. However, it remains challenging for users to retarget expressive motions onto these complex systems. In this paper, we present a *Differentiable Optimal Control* (DOC) framework that facilitates the transfer of rich motions from either animals or animations onto these robots. Interfacing with either motion capture or animation data, we formulate retargeting objectives whose parameters make them agnostic to differences in proportions and numbers of degrees of freedom between input and robot. Optimizing these parameters over the manifold spanned by optimal state and control trajectories, we minimize the retargeting error. We demonstrate the utility and efficacy of our modeling by applying DOC to a *Model-Predictive Control* (MPC) formulation, showing retargeting results for a family of robots of varying proportions and mass distribution. With a hardware deployment, we further show that the retargeted motions are physically feasible, while MPC ensures that the robots retain their capability to react to unexpected disturbances.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

*Joint first authors.

Authors' addresses: Ruben Grandia, ruben.grandia@disney.com, Disney Research, Switzerland; Farbod Farshidian, farshidian@mavt.ethz.ch, ETH Zürich, Switzerland; Espen Knoop, espen.knoop@disney.com, Disney Research, Switzerland; Christian Schumacher, christian.schumacher@disney.com, Disney Research, Switzerland; Marco Hutter, marco.hutter@ethz.ch, ETH Zürich, Switzerland; Moritz Bächer, moritz.baecher@disney.com, Disney Research, Switzerland.

0730-0301/2023/8-ART $15.00
https://doi.org/1

Additional Key Words and Phrases: differentiable optimal control, differential dynamic programming, motion retargeting, model-predictive control

ACM Reference Format:
Ruben Grandia, Farbod Farshidian, Espen Knoop, Christian Schumacher, Marco Hutter, and Moritz Bächer. 2023. DOC: Differentiable Optimal Control for Retargeting Motions onto Legged Robots. *ACM Trans. Graph.* 42, 4 (August 2023), 14 pages. https://doi.org/1

1 INTRODUCTION

Legged robots have become widely accessible, and are designed to perform fast and dynamic motions. While demonstrations of expressive performances such as dancing exist [Ackerman 2021; Bi et al. 2018], it requires expert knowledge and manual trial-and-error to make these legged systems appear as believable characters rather than robots.

To give these mechanical systems traits of animals, motion capture systems are a valuable source of data. Skilled artists have also mastered the art of breathing life into digital characters. However, significant differences in proportions, mass distributions, and number of degrees of freedom make the retargeting of motions onto these systems an utmost challenging task. Further complicating matters, artistic input is not constrained to obey the laws of physics.

Recent advances in machine learning has made it possible for robots to learn agile and dynamic motor skills [Peng et al. 2020]. However, artists only have limited or indirect control of the resulting motion. Alternative retargeting techniques that interface either with artistic or captured input exist. However, they either require the robot to have similar proportions [Kang et al. 2022, 2021], to be fully actuated [Hoshyari et al. 2019], or that the animation is created with a rig that has the same degrees of freedom as the robot.

In this paper, we describe a technique that enables the *optimal* retargeting of expressive motions onto freely walking robots. At the technical core of our approach is a *Differentiable Optimal Control (DOC)* framework that interfaces with arbitrary sources of motion data. While we ask users to define loose correspondences between points of interest on the input and the robot, we optimize parameters of a set of retargeting objectives to account for differences in proportions and degrees of freedom.

To demonstrate the utility of our modeling, we make a specific Model-Predictive Control (MPC) formulation [Grandia et al. 2022] differentiable with respect to objective parameters, then use it to transfer the same motion capture or animation data onto quadrupeds of vastly different proportions and mass distribution (see Fig. 1). Because the robots have to fulfill kinematic and dynamic constraints dictated by physics, the retargeting result is natural, with potential applications in physics-based character animation. As we show with experiments on a physical robot, the system retains its capability to react to uncertainty in the environment through online MPC, closely following the target motion before and after unexpected disturbances.

In short, our contributions are

- a differentiable *continuous* optimal control framework that applies to general *constrained* optimal control strategies, and
- an easy-to-use retargeting technique that is agnostic to differences in proportions, mass distributions, and degrees of freedom between the input source and the robot model.

2 RELATED WORK

Creating Realistic Motions. In computer graphics, much work has been dedicated to producing realistic-looking quadrupedal animations [Raibert and Hodgins 1991; Skrba et al. 2009]. Physics-based simulation techniques bring these results closer to the physical world [Coros et al. 2011; Ye and Liu 2010], and more recent work has also leveraged learning-based approaches to improve animations based on real-world data [Lee et al. 2021; Peng et al. 2022; Won et al. 2022; Yao et al. 2022a]. This is a valuable inspiration for robotic motions, but is not directly applicable.

For animated characters, some work has also looked at motion retargeting onto characters of different proportions [Reveret et al. 2005; Won and Lee 2019] or morphologies [Gleicher 1998; Hecker et al. 2008]. Da Silva et al. [2008] use an MPC formulation to track a mocap reference. Ryu et al. [2021] use a muscle-based retargeting for improved realism on animated characters. Zordan et al. [2002] combine a mocap reference with a physics simulation model to create responsive motions based on a mocap input. However, bringing motions onto physical robots presents additional challenges.

For humanoid robots, the problem of motion retargeting from a mocap input has been studied through model-based approaches [Ayusawa and Yoshida 2017; Dariush et al. 2008a; Darvish et al. 2019; Nakaoka et al. 2007; Pollard et al. 2002; Yamane et al. 2010], or by defining motion primitives [Dariush et al. 2008b; Ott et al. 2008]. Data-driven approaches have also been explored [Hausman et al. 2018; Sok et al. 2007]. Gielniak et al. [2011] evaluate metrics for measuring human-to-robot motion similarity. Retargeting problems have also been studied for humanoid robots for teleoperation tasks [Rouxel et al. 2022; Tosun et al. 2014], gestures [Choi et al. 2020, 2021], and locomotion [Taylor et al. 2021]. However, none of the above works demonstrate retargeting of highly dynamic locomotion with aperiodic footfall patterns.

Outside the domain of dynamic walking, Schumacher et al. [2021] consider motion retargeting for quasi-static walking, and Hoshyari et al. [2019] retarget dynamic motions for fixed-base robots. Gielniak et al. [2010] add so-called secondary animations to robot motions for improved realism.

Kim et al. [2022] demonstrate an early result of using a deep-learning-based formulation to retarget motions from a human in a mocap space onto a quadrupedal robot.

Creating Motions for Walking Robots. Recent years have seen a surge of work in the domain of quadrupedal locomotion, spurred by the availability of quadrupedal robots.

Model-based approaches to robot locomotion typically implement a Model Predictive Controller (MPC) [Farshidian et al. 2017a; Neunert et al. 2016]. By repeatedly solving an optimal control problem, functional motions such as walking can be generated [Apgar et al. 2018; Gehring et al. 2016; Grandia et al. 2019; Katz et al. 2019; Neunert et al. 2017; Nishiwaki et al. 2002; Xi and Remy 2014; Zhou et al. 2022]. These controllers are typically set up to follow hig-level walking commands and use handcrafted heuristics to regularize the walking style. During the hardware experiments in this work, we use MPC to track retargeted motions, removing the need for additional heuristics.

There are some recent examples where mocap or artistic input is combined with model-based locomotion approaches. Kang et al. [2022; 2021] extract the body and footfall pattern from a mocap source, then compute a robot gait using heuristics. Li et al. [2021] use trajectory optimization with a MPC-based approach to produce a robot gait which is close to a mocap input. In both of these works, the retargeting from mocap skeleton to robot was performed manually, and the transfer of motions onto robots with different proportions was not considered. By optimizing over retargeting parameters, we automate this task for a range of proportions and mass distributions.

Similar to the bi-level optimization approach proposed in this work, Zhao et al. [2020] solve a robot design optimization problem, wrapped around an MPC formulation. However, their focus is on optimizing the design of the robot rather than optimally retargeting a motion onto a given robot.

Boston Dynamics have reached world-wide fame with videos of dancing quadrupeds [2021], however their approach is proprietary and unpublished. In a blogpost, they briefly describe their "Choreographer" tool, which allows for the specification of dance sequences, and can also interface with animation data from Autodesk Maya, but their underlying approach is undisclosed.

Another branch of work has applied deep reinforcement learning and other machine learning methods to locomotion problems [Smith et al. 2021; Tan et al. 2018; Yang et al. 2020]. Recent results have demonstrated the learning of quadrupedal gaits from mocap sources on simulated robots [Yin et al. 2021] and on hardware [Bohez et al. 2022; Peng et al. 2020; Yao et al. 2022b]. While learning-based techniques can yield impressive performance, they commonly require

significant reward engineering if proportions and mass distributions between input and target robot differ.

Differential Dynamic Programming. DOC is closely related to Differential Dynamic Programming (DDP). Seminal work focused on unconstrained *discrete-time* optimal control problems [Jacobson and Mayne 1970; Mayne 1966; Mitter 1966], followed by formulations that can handle linear [Murray and Yakowitz 1979; Shi et al. 1990], box inequality [Tassa et al. 2014], or more general equality and inequality constraints [Aoyama et al. 2021; De O. Pantoja and Mayne 1989; Giftthaler and Buchli 2017; Howell et al. 2019; Lin and Arora 1991; Todorov and Li 2005; Xie et al. 2017]. We focus on a *continuous-time* formulation [Sleiman et al. 2021; Sun et al. 2014] and use projection to handle equality constraints [Farshidian et al. 2017b], similar to the QR-decomposition approach for state-only constraints described by Giftthaler et al. [2017].

However, in contrast, we not only solve an optimal control problem, but make the resulting optimal trajectories differentiable with respect to parameters. Oshin et al. [2022] describe a differentiable unconstrained discrete-time formulation, and Amos et al. [2018] present a similar method that also supports box constraints. Our technique interfaces with the larger class of equality-constrained Optimal Control (OC) problems. Moreover, we show that two similar sets of matrix Riccati equations can be derived to efficiently solve for optimal trajectories as well as their sensitivities with respect to a generic set of parameters, reducing the code that needs to be written to make an OC problem differentiable.

3 OVERVIEW

Before describing how we take derivatives of optimal control and robot state trajectories, we briefly review the continuous control problem considered here.

To get robots to perform dynamic motions, control problems of the form

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \int_{t_s}^{t_e} f(\mathbf{x}(t),\mathbf{u}(t);\mathbf{p})\,\mathrm{d}t \;+\; F(\mathbf{x}(t_e);\mathbf{p}) \qquad (1)$$
$$\text{s.t.}\quad \text{robot in dynamic equilibrium for all } t,$$

where $\mathbf{x}(t)$ is the time-varying state of the robot and $\mathbf{u}(t)$ the corresponding control trajectory, are solved for a fixed time horizon $[t_s, t_e]$. Because we assume a fixed and finite time horizon, it is common practice to differentiate between an intermediate objective f and a terminal objective F [Jacobson and Mayne 1970]. The intermediate objective measures the difference to the target motion, parameterized with parameters $\mathbf{p}$ that provide control of the retargeting result. The terminal objective accounts for effects beyond the finite horizon. For example, we can use F to ask for an end state from which the robot can transition into standing. We also add regularization terms to the objective pair, (f, F), where necessary.

With a motion capture system, we can track the motion of a set of points on an animal's body. Similarly, we can extract target trajectories of points of interest on an artist-specified input (Fig. 2, Input). To retarget an extracted motion onto a robot, it is natural to define reference frames on the robot, then guide their motion using the extracted target trajectories (Defining Correspondences). Measuring the differences between simulated and target states with

Algorithm 1 Optimal Control (OC)

function OC($\mathbf{p}, \mathbf{x}(t), \mathbf{u}(t)$)
  loop
    $\delta\mathbf{x}(t), \delta\mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\nu}(t) :=$ OCSearchDir($\mathbf{p}, \mathbf{x}(t), \mathbf{u}(t)$)
    find step length α [Nocedal and Wright 2006]
    $\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} := \begin{bmatrix}\mathbf{x}(t)\\ \mathbf{u}(t)\end{bmatrix} + \alpha \begin{bmatrix}\delta\mathbf{x}(t)\\ \delta\mathbf{u}(t)\end{bmatrix}$
    if converged **then** break **end if**
  end loop
  return $\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\nu}(t)$ ▷ optimal trajectories
end function

per-point objectives, we solve for optimal state and control trajectories, $\underline{\mathbf{x}}(t)$ and $\underline{\mathbf{u}}(t)$, that minimize a weighted sum of differences. We will describe our retargeting objectives in Sec. 6.

Because our objective pair and the robot dynamics are both, in general, nonlinear, we resort to an iterative scheme. However, in contrast to standard numerical minimization problems, the unknowns in our OC problem are time-varying functions instead of variables. Our search direction are therefore functions, $\delta\mathbf{x}(t)$ and $\delta\mathbf{u}(t)$, that we compute with a local approximation as we describe in Sec. 4. We can then perform line search to determine a step length, α, to update our current best estimate (see Alg. 1).

3.1 Defining the Retargeting Task

To make our retargeting agnostic to differences in proportions, mass distribution, and number of degrees of freedom, we parameterize, for example, the non-uniform scaling of target trajectories and the reference location and orientation of points of interest on the components of the robot (see Sec. 6).

Due to the limited number of degrees of freedom of a legged robot, the motion of its components is tightly coupled. Moreover, the robot will rarely have similar proportions and mass distribution to the input. It is therefore tedious for users to find good values for $\mathbf{p}$, and hence best to solve for optimal values using optimization.

3.2 Solving for Optimal Parameters

If we make adjustments to parameters, the optimal control and state trajectories change. We therefore need to solve the OC problem repeatedly when solving for optimal parameters. To this end, we formulate a bi-level or nested optimization where we solve for an optimal $\mathbf{p}$ in the outer loop (Alg. 2; Fig. 2, DOC), and for optimal trajectories in the inner loop (Alg. 1).

The outer optimization minimizes a pair of intermediate and terminal objectives, (g, G), that depend on optimal state and control trajectories, $\underline{\mathbf{x}}(t)$ and $\underline{\mathbf{u}}(t)$, and is set to a modified set of retargeting objectives and regularization terms

$$\min_{\mathbf{p}} \int_{t_s}^{t_e} g(\underline{\mathbf{x}}(t,\mathbf{p}),\underline{\mathbf{u}}(t,\mathbf{p});\mathbf{p})\,\mathrm{d}t \;+\; G(\underline{\mathbf{x}}(t_e,\mathbf{p});\mathbf{p}) \qquad (2)$$
$$\text{s.t.}\quad \text{parameters fulfill a set of requirements.}$$

Because the optimal trajectories *implicitly* depend on $\mathbf{p}$, we solve the OC problem whenever we evaluate the objective pair or its

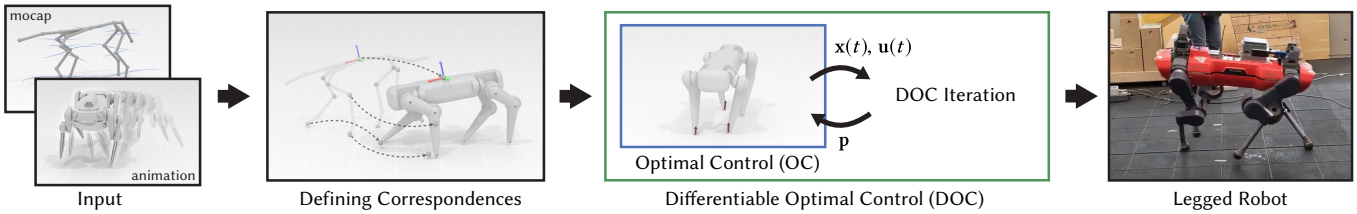

Fig. 2. **Overview.** Our processing starts with extracting motion trajectories of points of interest from captured input or common animation representations (Input). To define the retargeting problem, the user then selects corresponding locations on the robot (Defining Correspondences). Using our differentiable optimal control framework, we then make automated adjustments to the scaling of target trajectories or the locations of initial reference locations (DOC). Applying DOC to MPC results in expressive and believable motions on physical robots, while they retain their capability to respond and recover from external disturbances (Legged Robot).

Algorithm 2 Differentiable Optimal Control (DOC)

function DOC($\mathbf{p}, \mathbf{x}(t), \mathbf{u}(t)$)
  loop
    $\underline{\mathbf{x}}(t), \underline{\mathbf{u}}(t), \underline{\boldsymbol{\lambda}}(t), \underline{\boldsymbol{\nu}}(t) :=$ OC($\mathbf{p}, \mathbf{x}(t), \mathbf{u}(t)$)
    $\mathbf{x}_\mathbf{p}(t), \mathbf{u}_\mathbf{p}(t) :=$ DOCSens($\mathbf{p}, \underline{\mathbf{x}}(t), \underline{\mathbf{u}}(t), \underline{\boldsymbol{\lambda}}(t), \underline{\boldsymbol{\nu}}(t)$)
    evaluate gradient (Eq. 3)
    compute quasi-Newton search direction $\mathbf{d}$
    find step length α ([Nocedal and Wright 2006])
    $\mathbf{p} := \mathbf{p} - \alpha\,\mathbf{d}$
    $\mathbf{x}(t), \mathbf{u}(t) := \underline{\mathbf{x}}(t), \underline{\mathbf{u}}(t)$
    if converged **then** break **end if**
  end loop
  return $\mathbf{p}, \underline{\mathbf{x}}(t), \underline{\mathbf{u}}(t)$ ▷ optimal retargeting parameters
end function

gradient

$$\int_{t_s}^{t_e} \left(g_\mathbf{x}^T\, \mathbf{x}_\mathbf{p}(t) + g_\mathbf{u}^T\, \mathbf{u}_\mathbf{p}(t) + g_\mathbf{p}^T \right) \mathrm{d}t + G_\mathbf{x}^T\, \mathbf{x}_\mathbf{p}(t_e) + G_\mathbf{p}^T, \tag{3}$$

omitting optimal trajectory arguments. While the gradients of the intermediate and terminal functions with respect to state and control parameters ($g_\mathbf{x}$, $g_\mathbf{u}$, $g_\mathbf{p}$, $G_\mathbf{x}$ and $G_\mathbf{p}$) are straightforward to compute, the total derivatives of the optimal state and control trajectories, $\mathbf{x}_\mathbf{p}(t)$ and $\mathbf{u}_\mathbf{p}(t)$, cannot directly be computed using symbolic or automatic differentiation.

One of our technical contributions is the derivation of continuous two-boundary problems that allow us to compute search directions for our OC and sensitivities for our DOC problems (Sec. 4). To solve them, we propose a projection technique that results in efficient algorithms for the OCSearchDir and DOCSens functions (Sec. 5). We then apply DOC to Model-Predictive Control (MPC) in Sec. 6.

4 DIFFERENTIABLE OPTIMAL CONTROL

Optimal control of legged systems requires us to consider dynamic equilibrium and other algebraic constraints. We will first describe these constraints, recasting our optimal control problem as a non-linear two-boundary problem that we iteratively solve using the algorithm outlined in Alg. 1. We will then derive a second two-boundary problem to compute sensitivities in Alg. 2.

4.1 Optimally Controlling Legged Systems

To ensure that the optimal state and control trajectories are feasible, we constrain them to fulfill kinematic and dynamic constraints, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$. By introducing velocity variables, second-order equations of motion can always be brought into this standard form. The state $\mathbf{x}(t_s) = \mathbf{x}_s$ at the start time t_s is assumed to be known. Contacts between the feet and the ground form additional, time-varying, constraints on the feasible trajectories. We enforce these conditions with general equality constraints, $\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{0}$.

Moreover, we assume that inequality constraints, in our case friction cone constraints and joint and torque limits, are enforced with penalty functions that are part of the objective f [Grandia et al. 2022]. The continuous-time, equality constrained optimal control problem we consider is therefore

$$\begin{aligned} \min_{\mathbf{x}(t),\mathbf{u}(t)} & \int_{t_s}^{t_e} f(\mathbf{x}(t), \mathbf{u}(t); \mathbf{p})\, \mathrm{d}t \;+\; F(\mathbf{x}(t_e); \mathbf{p}) \\ \text{s.t.} \quad & \mathbf{x}(t_s) = \mathbf{x}_s \quad \text{and} \quad \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ & \text{and} \quad \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{0}. \end{aligned} \tag{4}$$

4.2 Solving for Optimal Trajectories

Common practice in deriving solution strategies for optimal control problems is to recast the minimization as a two-boundary problem [Jacobson and Mayne 1970]. Following standard procedure, we form the Lagrangian

$$L(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu}; \mathbf{p}) = \int_{t_s}^{t_e} H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu}; \mathbf{p}) - \boldsymbol{\lambda}^T \dot{\mathbf{x}}\, \mathrm{d}t \tag{5}$$

with Hamiltonian

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu}; \mathbf{p}) = f(\mathbf{x}, \mathbf{u}; \mathbf{p}) + \boldsymbol{\lambda}^T \mathbf{f}(\mathbf{x}, \mathbf{u}) + \boldsymbol{\nu}^T \mathbf{g}(\mathbf{x}, \mathbf{u}), \tag{6}$$

where $\boldsymbol{\lambda}(t)$ are so-called costates and $\boldsymbol{\nu}(t)$ Lagrange multipliers. We then recast the optimal control problem as the two-boundary problem

$$\begin{bmatrix} H_\mathbf{x}(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}}; \mathbf{p}) \\ H_\mathbf{u}(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}}; \mathbf{p}) \\ H_{\boldsymbol{\lambda}}(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}}; \mathbf{p}) \\ H_{\boldsymbol{\nu}}(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}}; \mathbf{p}) \end{bmatrix} = \begin{bmatrix} -\dot{\underline{\boldsymbol{\lambda}}} \\ \mathbf{0} \\ \dot{\underline{\mathbf{x}}} \\ \mathbf{0} \end{bmatrix} \quad \text{with} \quad \begin{aligned} \underline{\mathbf{x}}(t_s) &= \mathbf{x}_s \\ \underline{\boldsymbol{\lambda}}(t_e) &= F_\mathbf{x}(\underline{\mathbf{x}}(t_e)), \end{aligned} \tag{7}$$

by applying the Euler-Lagrange equation or Pontryagin's Minimum Principle [Bertsekas 2012]. The solution of this two-boundary

problem is the set of optimal state, control, costate and Lagrange multiplier trajectories that minimize our OC objective.

Because the boundary condition at t_e depends on the optimal state $\underline{\mathbf{x}}$, we cannot directly solve this problem. Moreover, since the equations in Eq. 7 are generally nonlinear in the state and control trajectories, it is common practice to resort to an iterative scheme.

To numerically solve standard nonlinear systems of equations, we linearize the equations around the current iterate, compute a search direction, and perform line search to determine the next iterate.

The setting here is different in that the unknowns are time-varying *functions* instead of variables. However, to solve the two-boundary problem, we can proceed analogously: We form first-order Taylor expansions of the right- and left-hand side of the nonlinear equations in problem 7,

$$\begin{bmatrix} H_{\mathbf{x}} \\ H_{\mathbf{u}} \\ H_{\boldsymbol{\lambda}} \\ H_{\boldsymbol{\nu}} \end{bmatrix} + \underbrace{\begin{bmatrix} H_{\mathbf{xx}} & H_{\mathbf{xu}} & H_{\mathbf{x}\boldsymbol{\lambda}} & H_{\mathbf{x}\boldsymbol{\nu}} \\ H_{\mathbf{ux}} & H_{\mathbf{uu}} & H_{\mathbf{u}\boldsymbol{\lambda}} & H_{\mathbf{u}\boldsymbol{\nu}} \\ H_{\boldsymbol{\lambda}\mathbf{x}} & H_{\boldsymbol{\lambda}\mathbf{u}} & H_{\boldsymbol{\lambda}\boldsymbol{\lambda}} & H_{\boldsymbol{\lambda}\boldsymbol{\nu}} \\ H_{\boldsymbol{\nu}\mathbf{x}} & H_{\boldsymbol{\nu}\mathbf{u}} & H_{\boldsymbol{\nu}\boldsymbol{\lambda}} & H_{\boldsymbol{\nu}\boldsymbol{\nu}} \end{bmatrix}}_{\boldsymbol{\mathcal{H}}} \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{u} \\ \boldsymbol{\lambda} \\ \boldsymbol{\nu} \end{bmatrix} \text{ and } \begin{bmatrix} -\dot{\boldsymbol{\lambda}} \\ \mathbf{0} \\ \dot{\mathbf{x}} + \delta\dot{\mathbf{x}} \\ \mathbf{0} \end{bmatrix}, \tag{8}$$

and the boundary condition at t_e

$$F_{\mathbf{x}}(\mathbf{x}) + F_{\mathbf{xx}}(\mathbf{x})\delta\mathbf{x}. \tag{9}$$

Because the Hamiltonian is already linear in co-states and Lagrange multipliers, we only linearize state and control trajectories in the above expansions, evaluating them at the point $(\mathbf{x}, \mathbf{u}, \mathbf{0}, \mathbf{0})$ in the neighborhood $(\delta\mathbf{x}, \delta\mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu})$. By substituting these three expansions in problem 7, and using the constraint $\dot{\mathbf{x}} = \mathbf{f}$ to simplify the third equation, we form the *linear* two-boundary problem

$$\begin{bmatrix} H_{\mathbf{xx}} & H_{\mathbf{xu}} & \mathbf{f}_{\mathbf{x}}^T & \mathbf{g}_{\mathbf{x}}^T \\ H_{\mathbf{ux}} & H_{\mathbf{uu}} & \mathbf{f}_{\mathbf{u}}^T & \mathbf{g}_{\mathbf{u}}^T \\ \mathbf{f}_{\mathbf{x}} & \mathbf{f}_{\mathbf{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{g}_{\mathbf{x}} & \mathbf{g}_{\mathbf{u}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{u} \\ \boldsymbol{\lambda} \\ \boldsymbol{\nu} \end{bmatrix} + \begin{bmatrix} H_{\mathbf{x}} \\ H_{\mathbf{u}} \\ \mathbf{0} \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} -\dot{\boldsymbol{\lambda}} \\ \mathbf{0} \\ \delta\dot{\mathbf{x}} \\ \mathbf{0} \end{bmatrix} \tag{10}$$

with conditions

$$\delta\mathbf{x}(t_s) = \mathbf{0} \quad \text{and} \quad \boldsymbol{\lambda}(t_e) = F_{\mathbf{xx}}(\mathbf{x}(t_e))\,\delta\mathbf{x}(t_e) + F_{\mathbf{x}}(\mathbf{x}(t_e)). \tag{11}$$

Solving the linearized problem for delta trajectories, $\delta\mathbf{x}(t)$ and $\delta\mathbf{u}(t)$, we then update our current best state and control trajectory estimates, $\mathbf{x}(t)$ and $\mathbf{u}(t)$, as summarized in Alg. 1. This algorithm is known as Differential Dynamic Programming (DDP) [Mayne 1966]. Standard formulations, however, do not consider an additional set of equality constraints.

It remains to discuss how we can efficiently solve the linear two-boundary problem for the delta trajectories. Because a similar system will emerge when we discuss derivatives of optimal trajectories, we will discuss the solution strategy after introducing our differentiable control formulation. To keep the notation in the next section compact, we introduce the Hamiltonian matrix $\boldsymbol{\mathcal{H}}$ that collects all second derivatives of the Hamiltonian in the above Taylor expansion.

4.3 Computing Sensitivities

To derive a two-boundary problem for computing sensitivities, we again use first-order Taylor expansions of problem 7, but now considering a change in parameters, $\mathbf{p} + \delta\mathbf{p}$. Because H is a function of optimal trajectories *and* the set of parameters, and optimal trajectories implicitly depend on $\mathbf{p}$, we expand the total derivative on either side of the equation in problem 7

$$\begin{bmatrix} H_{\mathbf{x}} \\ H_{\mathbf{u}} \\ H_{\boldsymbol{\lambda}} \\ H_{\boldsymbol{\nu}} \end{bmatrix} + \boldsymbol{\mathcal{H}} \begin{bmatrix} \mathbf{x}_{\mathbf{p}} \\ \mathbf{u}_{\mathbf{p}} \\ \boldsymbol{\lambda}_{\mathbf{p}} \\ \boldsymbol{\nu}_{\mathbf{p}} \end{bmatrix} \delta\mathbf{p} + \begin{bmatrix} H_{\mathbf{xp}} \\ H_{\mathbf{up}} \\ H_{\boldsymbol{\lambda}\mathbf{p}} \\ H_{\boldsymbol{\nu}\mathbf{p}} \end{bmatrix} \delta\mathbf{p} = \begin{bmatrix} -(\dot{\underline{\boldsymbol{\lambda}}} + \dot{\boldsymbol{\lambda}}_{\mathbf{p}}\delta\mathbf{p}) \\ \mathbf{0} \\ \dot{\underline{\mathbf{x}}} + \dot{\mathbf{x}}_{\mathbf{p}}\delta\mathbf{p} \\ \mathbf{0} \end{bmatrix}, \tag{12}$$

using time derivatives of expansions of optimal trajectories, for example, $\dot{\underline{\mathbf{x}}}(t, \mathbf{p} + \delta\mathbf{p}) \approx \dot{\underline{\mathbf{x}}}(t, \mathbf{p}) + \dot{\mathbf{x}}_{\mathbf{p}}(t)\,\delta\mathbf{p}$, for state trajectories, on the right-hand side. Because $H_{\mathbf{x}} = -\dot{\underline{\boldsymbol{\lambda}}}$, $H_{\mathbf{u}} = \mathbf{0}$, $H_{\boldsymbol{\lambda}} = \dot{\underline{\mathbf{x}}}$, $H_{\boldsymbol{\nu}} = \mathbf{0}$ at $(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}})$, and $\delta\mathbf{p}$ cancels, we can bring these equations into standard form

$$\begin{bmatrix} H_{\mathbf{xx}} & H_{\mathbf{xu}} & \mathbf{f}_{\mathbf{x}}^T & \mathbf{g}_{\mathbf{x}}^T \\ H_{\mathbf{ux}} & H_{\mathbf{uu}} & \mathbf{f}_{\mathbf{u}}^T & \mathbf{g}_{\mathbf{u}}^T \\ \mathbf{f}_{\mathbf{x}} & \mathbf{f}_{\mathbf{u}} & \mathbf{0} & \mathbf{0} \\ \mathbf{g}_{\mathbf{x}} & \mathbf{g}_{\mathbf{u}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathbf{p}} \\ \mathbf{u}_{\mathbf{p}} \\ \boldsymbol{\lambda}_{\mathbf{p}} \\ \boldsymbol{\nu}_{\mathbf{p}} \end{bmatrix} + \begin{bmatrix} H_{\mathbf{xp}} \\ H_{\mathbf{up}} \\ \mathbf{f}_{\mathbf{p}} \\ \mathbf{g}_{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} -\dot{\boldsymbol{\lambda}}_{\mathbf{p}} \\ \mathbf{0} \\ \dot{\mathbf{x}}_{\mathbf{p}} \\ \mathbf{0} \end{bmatrix}, \tag{13}$$

with boundary conditions

$$\mathbf{x}_{\mathbf{p}}(t_s) = \mathbf{0} \quad \text{and} \quad \boldsymbol{\lambda}_{\mathbf{p}}(t_e) = F_{\mathbf{xx}}(\underline{\mathbf{x}}(t_e))\mathbf{x}_{\mathbf{p}}(t_e) + F_{\mathbf{xp}}(\underline{\mathbf{x}}(t_e)). \tag{14}$$

Note that the resulting derivatives are *exact* because the resulting system is independent of $\delta\mathbf{p}$. This is also the case if we use a higher-order Taylor expansion, providing a recipe to compute higher-order derivatives. All Hamiltonian derivatives are evaluated at optimal trajectories, $(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}})$, for a given $\mathbf{p}$.

A key insight is that the above linear two-boundary problem resembles the linearized problem 10 we solve when iteratively computing optimal trajectories. We can therefore use the same solution strategy for solving both linear two-boundary problems, as we will explain in the next section.

5 SOLVING LINEAR TWO-BOUNDARY PROBLEMS

For problems without equality constraints, a common solution strategy for linear two-boundary problems is the so-called backward sweep method [Bryson Jr 1965; Mitter 1966]. In this method, one of the two boundary conditions is removed by recasting the problem. The resulting set of matrix Riccati equations is then solved backward in time.

What is different in our setting is that we have a set of equality constraints, and therefore a fourth equation and an additional set of unknown Lagrange multiplier trajectories. Below, we describe a technique to project the continuous equations onto the constraint manifold, reducing the 4x4 to a 2x2 two-boundary problem that we know how to solve using Riccati equations [Bertsekas 2012]. We describe our projection next, then derive the Riccati equations in Sec. 5.2

5.1 Projecting Equations onto Constraint Manifold

The linear two-boundary OC problem 10 and DOC problem 13 are similar. The differences are

- In the OC problem, derivatives are evaluated at the current iterate $(\mathbf{x}, \mathbf{u}, \mathbf{0}, \mathbf{0})$, while we use optimal trajectories, $(\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}})$, as arguments in the DOC problem.

- The unknowns in the OC problem are time-varying vector functions $(\delta\mathbf{x}, \delta\mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, while we solve for time-varying matrix functions, $(\mathbf{x_p}, \mathbf{u_p}, \boldsymbol{\lambda_p}, \boldsymbol{\nu_p})$ in the DOC problem.
- The constant vector in the inhomogeneous linear system of equations has entries $(H_\mathbf{x}, H_\mathbf{u}, \mathbf{0}, \mathbf{g})$ in the OC, and entries $(H_\mathbf{xp}, H_\mathbf{up}, \mathbf{f_p}, \mathbf{g_p})$ in the DOC problem.
- In the boundary conditions for co-states at time t_e, we are using terminal function derivatives $(F_\mathbf{xx}, F_\mathbf{x})$ in the OC, and derivatives $(F_\mathbf{xx}, F_\mathbf{xp})$ in the DOC problem.

To derive our solution strategy, we work with time-varying vector variables $(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, a constant vector $\boldsymbol{C}$ with entries $(H_\mathbf{x}, H_\mathbf{u}, \mathbf{f}, \mathbf{g})$, and terminal function derivatives $(F_\mathbf{xx}, F_\mathbf{x})$ for which we can substitute quantities for the OC and DOC problems to derive our final algorithms:

$$\begin{bmatrix} H_\mathbf{xx} & H_\mathbf{xu} & \mathbf{f}_\mathbf{x}^T & \mathbf{g}_\mathbf{x}^T \\ H_\mathbf{ux} & H_\mathbf{uu} & \mathbf{f}_\mathbf{u}^T & \mathbf{g}_\mathbf{u}^T \\ \mathbf{f_x} & \mathbf{f_u} & \mathbf{0} & \mathbf{0} \\ \mathbf{g_x} & \mathbf{g_u} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \boldsymbol{\lambda} \\ \boldsymbol{\nu} \end{bmatrix} + \begin{bmatrix} H_\mathbf{x} \\ H_\mathbf{u} \\ \mathbf{f} \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} -\dot{\boldsymbol{\lambda}} \\ \mathbf{0} \\ \dot{\mathbf{x}} \\ \mathbf{0} \end{bmatrix} \tag{15}$$

with boundary conditions

$$\mathbf{x}(t_s) = \mathbf{0} \quad \text{and} \quad \boldsymbol{\lambda}(t_e) = F_\mathbf{xx}(\mathbf{x}(t_e))\mathbf{x}(t_e) + F_\mathbf{x}(\mathbf{x}(t_e)). \tag{16}$$

Our goal is to first remove the constraint equation and the Lagrange multiplier trajectories from the above system, reducing it to a 3x3 problem. To do so, we work with a reduced set of control variables that fulfill the constraint equations.

Assuming that the constraint Jacobian $\mathbf{g_u}$ has full row rank, we form the QR decomposition of its transpose

$$\mathbf{g}_\mathbf{u}^T = \mathbf{QR} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix} = \mathbf{Q}_1\mathbf{R}_1. \tag{17}$$

We then substitute $\mathbf{R}_1^T\mathbf{Q}_1^T$ for $\mathbf{g_u}$ in the 4th equation to express the set of all control trajectories that satisfy the constraint equations as a combination of a range space and null space solution that is parameterized with subspace control trajectories $\tilde{\mathbf{u}}(t)$

$$\mathbf{u} = \mathbf{P_x}\mathbf{x} + \mathbf{P_u}\tilde{\mathbf{u}} + \mathbf{P} \quad \text{with} \tag{18}$$

$$\mathbf{P_x} = -\mathbf{Q}_1\mathbf{R}_1^{-T}\mathbf{g_x}, \quad \mathbf{P_u} = \mathbf{Q}_2, \quad \mathbf{P} = -\mathbf{Q}_1\mathbf{R}_1^{-T}\mathbf{g}. \tag{19}$$

Substituting for $\mathbf{u}$ in problem 15, we remove the direct dependence on control trajectories

$$\begin{bmatrix} H_\mathbf{xx} + H_\mathbf{xu}\mathbf{P_x} & H_\mathbf{xu}\mathbf{P_u} & \mathbf{f}_\mathbf{x}^T & \mathbf{g}_\mathbf{x}^T \\ H_\mathbf{ux} + H_\mathbf{uu}\mathbf{P_x} & H_\mathbf{uu}\mathbf{P_u} & \mathbf{f}_\mathbf{u}^T & \mathbf{g}_\mathbf{u}^T \\ \mathbf{f_x} + \mathbf{f_u}\mathbf{P_x} & \mathbf{f_u}\mathbf{P_u} & \mathbf{0} & \mathbf{0} \\ \mathbf{g_x} + \mathbf{g_u}\mathbf{P_x} & \mathbf{g_u}\mathbf{P_u} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{u}} \\ \boldsymbol{\lambda} \\ \boldsymbol{\nu} \end{bmatrix} + \begin{bmatrix} H_\mathbf{x} + H_\mathbf{xu}\mathbf{P} \\ H_\mathbf{u} + H_\mathbf{uu}\mathbf{P} \\ \mathbf{f} + \mathbf{f_u}\mathbf{P} \\ \mathbf{g} + \mathbf{g_u}\mathbf{P} \end{bmatrix} = \begin{bmatrix} -\dot{\boldsymbol{\lambda}} \\ \mathbf{0} \\ \dot{\mathbf{x}} \\ \mathbf{0} \end{bmatrix}.$$

By using properties of the QR decomposition, we can easily verify that all coefficients in the last equation, namely $\mathbf{g_x} + \mathbf{g_u}\mathbf{P_x}$, $\mathbf{g_u}\mathbf{P_u}$, and $\mathbf{g} + \mathbf{g_u}\mathbf{P}$, are zero.

By premultiplying the equations with the projection matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{P}_\mathbf{x}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_\mathbf{u}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{20}$$

the coefficients corresponding to the Lagrange multiplier trajectory become zero, resulting in the system

$$\left[\begin{array}{ccc|c} \tilde{H}_\mathbf{xx} & \tilde{H}_\mathbf{xu} & \tilde{\mathbf{f}}_\mathbf{x}^T & \mathbf{0} \\ \tilde{H}_\mathbf{ux} & \tilde{H}_\mathbf{uu} & \tilde{\mathbf{f}}_\mathbf{u}^T & \mathbf{0} \\ \tilde{\mathbf{f}}_\mathbf{x} & \tilde{\mathbf{f}}_\mathbf{u} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right] \left[\begin{array}{c} \mathbf{x} \\ \tilde{\mathbf{u}} \\ \boldsymbol{\lambda} \\ \hline \boldsymbol{\nu} \end{array}\right] + \left[\begin{array}{c} \tilde{H}_\mathbf{x} \\ \tilde{H}_\mathbf{u} \\ \tilde{\mathbf{f}} \\ \hline \mathbf{0} \end{array}\right] = \left[\begin{array}{c} -\dot{\boldsymbol{\lambda}} \\ \mathbf{0} \\ \dot{\mathbf{x}} \\ \hline \mathbf{0} \end{array}\right] \tag{21}$$

whose solution is independent of the 4th equation and the Lagrange multipier trajectories. Explicit expressions for the entries of the reduced 3x3 Hamiltonian matrix $\tilde{\boldsymbol{\mathcal{H}}}$ as well as the entries of the 3-entry vector $\tilde{\boldsymbol{C}}$ can be found in the Appendix.

In a last reduction step, we remove the second equation by substituting

$$\tilde{\mathbf{u}} = -\tilde{H}_\mathbf{uu}^{-1}\left(\tilde{H}_\mathbf{ux}\mathbf{x} + \tilde{\mathbf{f}}_\mathbf{u}^T\boldsymbol{\lambda} + \tilde{H}_\mathbf{u}\right) \tag{22}$$

in the remaining two equations, resulting in the 2x2 system

$$\begin{bmatrix} \hat{H}_\mathbf{xx} & -\hat{\mathbf{f}}_\mathbf{x}^T \\ \hat{\mathbf{f}}_\mathbf{x} & \hat{\mathbf{H}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \hat{H}_\mathbf{x} \\ \hat{\mathbf{f}} \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\lambda}} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{23}$$

with the 2x2 Hamiltonian matrix $\hat{\boldsymbol{\mathcal{H}}}$ and 2-entry vector $\hat{\boldsymbol{C}}$ (see Appendix).

5.2 Reducing the Two- to a Single-Boundary Problem

A general two-boundary problem is difficult to solve, but because the two-boundary problem is linear, we can recast it as a single-boundary problem [Bertsekas 2012]. To this end, we assume that there is a matrix and vector, $\mathbf{S}(t)$ and $\mathbf{s}(t)$, such that

$$\boldsymbol{\lambda}(t) = \mathbf{S}(t)\mathbf{x}(t) + \mathbf{s}(t) \tag{24}$$

for all t. At time t_e, the boundary conditions are fulfilled if we set $\mathbf{S}(t_e)$ and $\mathbf{s}(t_e)$ to the two terminal functions, $F_\mathbf{xx}(\mathbf{x}(t_e))$ and $F_\mathbf{x}(\mathbf{x}(t_e))$, respectively.

If we substitute the above expression and its time derivative in the 2x2 two-boundary problem

$$\begin{bmatrix} \hat{H}_\mathbf{xx} & -\hat{\mathbf{f}}_\mathbf{x}^T \\ \hat{\mathbf{f}}_\mathbf{x} & \hat{\mathbf{H}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{S}\mathbf{x} + \mathbf{s} \end{bmatrix} + \begin{bmatrix} \hat{H}_\mathbf{x} \\ \hat{\mathbf{f}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{S}}\mathbf{x} + \mathbf{S}\dot{\mathbf{x}} + \dot{\mathbf{s}} \\ \dot{\mathbf{x}} \end{bmatrix},$$

then substitute the second equation

$$\dot{\mathbf{x}} = (\hat{\mathbf{f}}_\mathbf{x} + \hat{\mathbf{H}}\mathbf{S})\mathbf{x} + (\hat{\mathbf{f}} + \hat{\mathbf{H}}\mathbf{s}), \tag{25}$$

which represents the projected simulation, into the first one, we get

$$\left(\dot{\mathbf{S}} - \hat{H}_\mathbf{xx} + \hat{\mathbf{f}}_\mathbf{x}^T\mathbf{S} + \mathbf{S}\hat{\mathbf{f}}_\mathbf{x} + \mathbf{S}\hat{\mathbf{H}}\mathbf{S}\right)\mathbf{x} + \left(\dot{\mathbf{s}} - \hat{H}_\mathbf{x} + \hat{\mathbf{f}}_\mathbf{x}^T\mathbf{s} + \mathbf{S}\hat{\mathbf{f}} + \mathbf{S}\hat{\mathbf{H}}\mathbf{s}\right) = \mathbf{0}.$$

This equation can only be satisfied for general $\mathbf{x} \neq \mathbf{0}$ if

$$\begin{aligned} \dot{\mathbf{S}} &= \hat{H}_\mathbf{xx} - \hat{\mathbf{f}}_\mathbf{x}^T\mathbf{S} - \mathbf{S}\hat{\mathbf{f}}_\mathbf{x} - \mathbf{S}\hat{\mathbf{H}}\mathbf{S} \\ \dot{\mathbf{s}} &= \hat{H}_\mathbf{x} - \hat{\mathbf{f}}_\mathbf{x}^T\mathbf{s} - \mathbf{S}\hat{\mathbf{f}} - \mathbf{S}\hat{\mathbf{H}}\mathbf{s} \end{aligned} \tag{26}$$

with boundary conditions

$$\mathbf{S}(t_e) = F_\mathbf{xx}(\mathbf{x}(t_e)) \quad \text{and} \quad \mathbf{s}(t_e) = F_\mathbf{x}(\mathbf{x}(t_e)). \tag{27}$$

These so-called matrix Riccati equations [Bertsekas 2012] can then be solved backwards in time. See Alg. 3 for final algorithms to compute search directions for the OC and sensitivities for the DOC problem.

Algorithm 3 OC Search Direction and DOC Sensitivities

function TwoBoundaryProblem($\mathcal{P}, \boldsymbol{C}, \mathcal{F}$)
  for all t **do**
    evaluate matrix $\mathcal{H}$ and vector $\boldsymbol{C}$ at point $\mathcal{P}$
    evaluate terminal function derivatives $\mathcal{F}$ at point $\mathcal{P}$
    form QR decomposition of $\mathbf{g}_\mathbf{u}^T$ (Eq. 17)
    compute $\mathbf{P}_\mathbf{x}$, $\mathbf{P}_\mathbf{u}$, and $\mathbf{P}$ (Eq. 18)
    compute $\tilde{\mathcal{H}}$ and $\tilde{\boldsymbol{C}}$ (Eqs. 21, 32, 32)
    compute $\dot{\mathcal{H}}$ and $\dot{\boldsymbol{C}}$ (Eqs. 23, 34, 34)
  end for
  solve matrix Riccati eqs. backw. in time (Eqs. 26, 27)
  return $\mathbf{S}(t), \mathbf{s}(t)$
end function
function OCSearchDir($\mathbf{p}, \mathbf{x}(t), \mathbf{u}(t)$)
  $\mathcal{P} := (\mathbf{x}, \mathbf{u}, \mathbf{0}, \mathbf{0}), \boldsymbol{C} := (H_\mathbf{x}, H_\mathbf{u}, \mathbf{0}, \mathbf{g}), \mathcal{F} := (F_{\mathbf{xx}}, F_\mathbf{x})$
  $\mathbf{S}(t), \mathbf{s}(t) :=$ TwoBoundaryProblem($\mathcal{P}, \boldsymbol{C}, \mathcal{F}$)
  compute $\delta\mathbf{x}(t)$ by solving proj. sim. forw. in time (Eq. 25)
  compute $\delta\mathbf{u}(t)$ using Eqs. 18 and 22
  compute $\boldsymbol{\lambda}(t)$ using Eq. 24
  compute $\boldsymbol{\nu}(t)$ using the 2nd eq. in system 15
  return $\delta\mathbf{x}(t), \delta\mathbf{u}(t), \boldsymbol{\lambda}(t), \boldsymbol{\nu}(t)$
end function
function DOCSens($\mathbf{p}, \underline{\mathbf{x}}(t), \underline{\mathbf{u}}(t), \underline{\boldsymbol{\lambda}}(t), \underline{\boldsymbol{\nu}}(t)$)
  $\mathcal{P} := (\underline{\mathbf{x}}, \underline{\mathbf{u}}, \underline{\boldsymbol{\lambda}}, \underline{\boldsymbol{\nu}}), \boldsymbol{C} := (H_{\mathbf{xp}}, H_{\mathbf{up}}, \mathbf{g}_\mathbf{p}, \mathbf{g}_\mathbf{p}), \mathcal{F} := (F_{\mathbf{xx}}, F_{\mathbf{xp}})$
  $\mathbf{S}(t), \mathbf{s}(t) :=$ TwoBoundaryProblem($\mathcal{P}, \boldsymbol{C}, \mathcal{F}$)
  compute $\mathbf{x}_\mathbf{p}(t)$ by solving proj. sim. forw. in time (Eq. 25)
  compute $\mathbf{u}_\mathbf{p}(t)$ using Eqs. 18 and 22
  return $\mathbf{x}_\mathbf{p}(t), \mathbf{u}_\mathbf{p}(t)$
end function

6 APPLICATION: LEGGED ROBOTS

While DOC is applicable to a large class of control problems and is agnostic to the parameters that we seek to optimize, we apply our approach to a model-predictive control formulation for legged systems, solving an optimal retargeting task that enables the transfer onto robots of varying sizes, shapes, and mass distributions. Before discussing the MPC simulation model and the control parameters, we introduce our parameterized retargeting objectives, which are used in both the inner and outer loop of our bi-level optimization.

6.1 Retargeting Objectives

To direct legged systems, standard objectives control the base position and orientation of the robot, and the positions and velocities of its joints [Tassa et al. 2012]. To provide more flexiblity, we instead rely on parameterized objectives that measure differences in position, orientation, and linear and angular velocity at a discrete set of locations, letting our optimization decide on how to optimally retarget the motion.

When defining correspondences, a user selects a location, $\mathbf{r}_\text{rb}$, and orthonormal frame axes, $\mathbf{A}_\text{rb} = \left[\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z\right]$, in local coordinates of a robot's component for every point of interest. To measure differences to a provided target motion, we then transform the frame from local to global coordinates

$$\mathbf{r}(\mathbf{x}(t); \mathbf{p}) = \mathbf{R}\,\mathbf{r}_\text{rb} + \mathbf{t} \quad \text{and} \quad \mathbf{A}(\mathbf{x}(t); \mathbf{p}) = \mathbf{R}\,\mathbf{A}_\text{rb}, \tag{28}$$

where the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ depend on the state trajectory. The frame origin and its axes are part of the set of parameters $\mathbf{p}$ that we can optimize. This is helpful if, for example, two points of interest are guiding the motion of a *single* rigid component. DOC can refine initial reference frames to correct offsets between them.

Our first type of retargeting objective penalizes differences in *linear motion* between simulated and target positions, $\mathbf{r}$ and $\hat{\mathbf{r}}$, and corresponding velocities

$$\|\mathbf{r} - \mathbf{D}\hat{\mathbf{r}}\|^2_{\mathbf{W}_\mathbf{r}} + \|\dot{\mathbf{r}} - \mathbf{D}\dot{\hat{\mathbf{r}}}\|^2_{\mathbf{W}_{\dot{\mathbf{r}}}}, \tag{29}$$

with scaling factors, $\mathbf{D} = \text{diag}(s_{xy}, s_{xy}, s_z)$, that enable non-uniform scaling of target trajectories and can be included in the set of parameters. Note that we intentionally keep the scaling factors in the xy-plane, or more precisely the plane orthogonal to the direction of gravity, the same, because otherwise the motion would be warped if the robot walked in a non-axis-aligned direction.

We define an analogous type of objective for *angular motion*

$$\|\mathbf{A} \boxminus \hat{\mathbf{A}}\|^2_{\mathbf{W}_\mathbf{A}} + \|\boldsymbol{\omega} - \hat{\boldsymbol{\omega}}\|^2_{\mathbf{W}_{\boldsymbol{\omega}}}. \tag{30}$$

where $\boxminus$ is the logarithm map operator that measures the difference between the two rotation matrices with a 3D angle-axis vector, and $\boldsymbol{\omega}$ is the angular velocity of the component that frame $\mathbf{A}_\text{rb}$ is attached to. Note that $[\boldsymbol{\omega}]_\times = \dot{\mathbf{A}}\mathbf{A}^T = \dot{\mathbf{R}}\mathbf{R}^T$.

The weight matrices, $\mathbf{W}_\mathbf{r}$, $\mathbf{W}_{\dot{\mathbf{r}}}$, $\mathbf{W}_\mathbf{A}$, and $\mathbf{W}_{\boldsymbol{\omega}}$, are all 3D diagonal matrices and provide users with control of the relative importance of objectives and objective terms.

6.2 Model-Predictive Control

The following paragraphs describe the MPC formulation used for the optimal motion retargeting. Note that we consider the full time horizon of the input motion at once and start at a given initial state of the robot. The formulation follows the approach described in [Grandia et al. 2022], with the cost function adapted to the retargeting application.

Simulation Model. For simulation, we rely on a simplified centroidal dynamics formulation. We represent the state $\mathbf{x}$ of the robot with the position $\mathbf{c}$, orientation $\boldsymbol{\theta}$ (in Euler angles), linear velocity $\mathbf{v}$, and angular velocity $\boldsymbol{\omega}$, of its base, and the joint positions, $\mathbf{q}$, of its

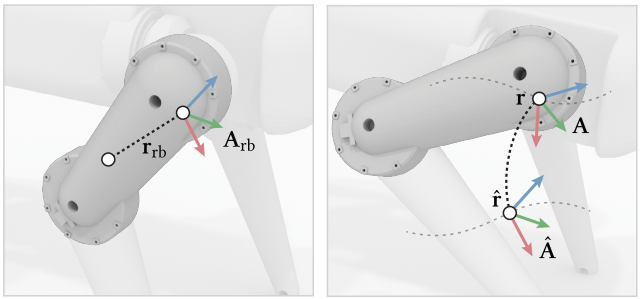

Fig. 3. **Retargeting Objectives.** A user selects a location $\mathbf{r}_\text{rb}$ and frame axes $\mathbf{A}_\text{rb}$ in local component coordinates (left). Our retargeting objectives then measure the differences between simulated positions and axes, $\mathbf{r}$ and $\mathbf{A}$, and their corresponding targets, $\hat{\mathbf{r}}$ and $\hat{\mathbf{A}}$, in global coordinates (right).

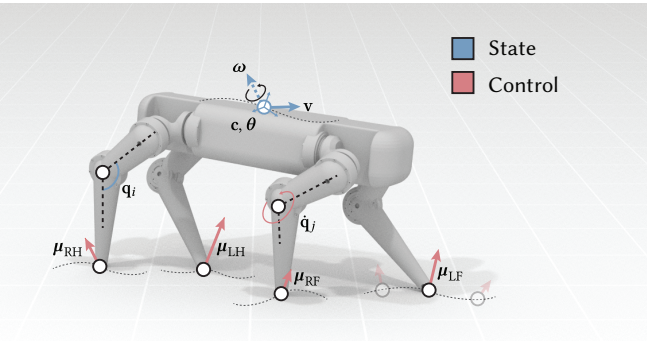

Fig. 4. **MPC** Simulation variables (in blue) and control parameters (in red).

actuators as shown in Fig. 4 in blue. The control parameters $\mathbf{u}$ that we consider are the forces $\boldsymbol{\mu}$ that act from the ground onto the feet, together with the joint velocities $\dot{\mathbf{q}}$ (in red). The centroidal dynamics

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{f}_1(\boldsymbol{\theta}, \mathbf{v}) \\ \mathbf{f}_2(\boldsymbol{\theta}, \boldsymbol{\omega}) \\ \mathbf{f}_3(\mathbf{x}, \mathbf{u}) \\ \mathbf{f}_4(\mathbf{x}, \mathbf{u}) \\ \dot{\mathbf{q}} \end{bmatrix}, \quad \text{with} \quad \mathbf{x} = \begin{bmatrix} \mathbf{c} \\ \boldsymbol{\theta} \\ \mathbf{v} \\ \boldsymbol{\omega} \\ \mathbf{q} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} \boldsymbol{\mu} \\ \dot{\mathbf{q}} \end{bmatrix}, \tag{31}$$

are governed by four nonlinear equations [Grandia et al. 2022]: $\mathbf{f}_1$ transforms the linear velocity of the base in local base to world coordinates, $\mathbf{f}_2$ uses a conversion between angular body rates and Euler angle derivatives to transform the angular velocity of the base, and $\mathbf{f}_2$ and $\mathbf{f}_3$ rely on an augmented centroidal dynamics that encourages smooth solutions for control inputs $\mathbf{u}(t)$.

Constraints. We enforce contact conditions with equality constraints, $\mathbf{g}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$. If a foot is in contact with the ground, its height and velocity has to remain zero, and if a foot is in a swing phase, the contact force that acts on the foot cannot take on non-zero values

$$\begin{aligned} &\mathbf{v}_i(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{0}, \quad \mathbf{h}_i(\mathbf{x}(t)) = \mathbf{0}, && \forall i \in \mathcal{I}(t), \\ &\boldsymbol{\mu}_i(\mathbf{u}(t)) = \mathbf{0}, && \forall i \notin \mathcal{I}(t), \end{aligned}$$

where $\mathbf{v}_i$, $\mathbf{h}_i$, and $\boldsymbol{\mu}_i$ are the linear velocity, foot height, and contact force for foot i, respectively. The subscripts for the legs that are in contact with the ground are contained in the set $\mathcal{I}$, which is a subset of all possible contact points, $\mathcal{I} \subseteq \{\text{LF}, \text{RF}, \text{LH}, \text{RH}\}$ (see Fig. 4). The velocity and contact force constraint are handled with the projection approach described in Sec. 5.1, and the foot height constraint is enforced through a quadratic penalty. To ensure that our control problem remains differentiable, we assume the footfall pattern to remain fixed and extract it from the input motion in a pre-processing step.

Objectives. For MPC, we set the intermediate and terminal objectives, f and F, to a weighted sum of our parameterized retargeting objectives, a set of penalties that enforce limits and friction constraints [Grandia et al. 2022], and regularization terms that penalize the magnitude of joint velocities and contact forces. For F, we add an optional term that minimizes the squared difference between $\mathbf{q}(t_e)$ and the nominal actuator positions, with a weight that varies per actuator.

Numerical Optimization. In proximity to optimal trajectories, the Hessian $\mathbf{H}_{\mathbf{uu}}$ is positive definite. However, for trajectories that are far from optimal, $\mathbf{H}_{\mathbf{uu}}$ is often indefinite, causing the iterative scheme (Alg. 1) to diverge [Mitter 1966]. A common strategy to mitigate this problem is to use an approximation of the Hessian where second derivatives of the dynamics and constraints are omitted when constructing the Hamiltonian matrix $\mathcal{H}$ [Farshidian et al. 2017b; Li and Todorov 2004]. This is similar to a Gauss-Newton scheme in the discrete setting.

6.3 Differentiable Model-Predictive Control

In our outer optimization, we set the intermediate and terminal objectives, g and G, to the same weighted sum of retargeting objectives, but omit penalties. However, we add a regularizer that penalizes the squared difference between $\mathbf{p}$ and their initial values to ensure that the DOC problem is well-posed, independent of the number of retargeting objectives.

Time Integration and Integrals. For time integration of the two boundary value problems, we use the Runge-Kutta-Fehlberg method (RKF45). To evaluate the integrals in objectives and their gradients, a zero-order hold approximation provides sufficient accuracy.

Search Direction. To obtain a search direction in Alg. 2, we use the Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method and corresponding line search [Nocedal and Wright 2006].

7 RESULTS

We evaluate our method by retargeting both motion capture data and artistic input animation onto a set of 10 different robots, thereby demonstrating generality to different inputs and across robots. We show that the retargeted motions can be executed on the physical robot, and that the robot retains its robustness to external disturbances while executing the motions. In the following, we first evaluate the effect of robot variations such as kinematics, dynamics, and body shape in the retargeting process. We then study the performance of the two-level optimization.

7.1 Motions and Robots

In our pipeline, we aim for automatic retargeting with minimum human intervention. We assume that the input target motion starts at the origin and we initialize the robot in a standing configuration with feet at zero height. The gait sequence, i.e., the footfall pattern, is extracted from the target using a threshold on the foot velocity. The user specifies pairs of frames from the target motion and frames attached to the robot, which define our objective function in the retargeting pipeline as introduced in Sec. 6.1.

Motions. We use a selection of motion capture sequences taken from the dataset in Zhang et al. [2018], which contains motion capture data of a dog for a set of different motions including locomotion, idling, and playful jumping and bounding. The dataset provides a skeletal animation for each motion, and we extract the locations of the rig joints for use as input to our pipeline. We also use a set of artist-created quadruped animations. These were created using standard animation tools, and do therefore not take into account

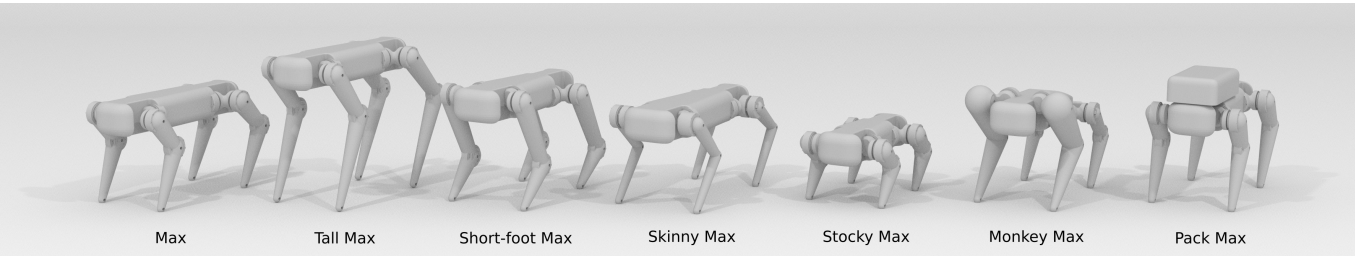

Fig. 5. **Max Family**: We retarget our motions onto a family of 7 simulated dog-like robots of varying sizes, proportions, and mass distributions.

Table 1. Motion capture (M) and animation (A) IDs together with short descriptions of the motions.

ID	Description
M1	High jump; tight turn after landing.
M2	Standing up from lying; stepping in-place, looking around.
M3	Explosive sprint; turn while looking down; walk away.
M4	Forward pace; looking around.
M5	Walking with tight turns, legs crossing over.
M6	Standing on box; stepping onto ground; jumping forwards.
M7	Jumping forwards; walking onto box.
M8	Walking around; turning; short gallop.
A1	Notices object and takes closer look; hopping/walking sideways.
A2	Jumping straight up from standing.
A3	Extending and swirling diagonal pairs of legs.
A4	Turning around, quick short steps.

the physics of the robot. See Table 1 for a list of the motion capture and animation motions.

Robots. To demonstrate that our method generalizes across robots, we show results on the following set, of varying size, proportions, and kinematics:

- A family of simulated dog-like robots (the "Max family"), with varying proportions (see Fig. 5): **Max** (dog-like size and proportions); **Tall Max** (taller); **Short-foot Max** (different leg proportions), **Skinny Max** (lighter legs, knee actuators in hip); **Stocky Max** (short and stubby); **Monkey Max** (longer fore-legs, shorter hind-legs); and **Pack Max** (carrying a heavy backpack).
- **Lizard**, a simulated robot with leg kinematics resembling those of a lizard, where the legs point outwards from the body (see Fig. 6, bottom).
- The **ANYmal** robot platform from ANYbotics, for which we also show a hardware result. We also show a simulated result for a fictitious **Strong ANYmal**, which has stronger actuators.

The simulated robots are between 300 and 605 mm tall, weigh between 16 and 27 kg, and have 3 actuators per leg. They have been designed so that they *could* be built in the future, having plausible geometry and mass distributions. We use velocity limits of 20 rad/s and torque limits of 200 Nm, which is in line with state-of-the-art actuators for quadrupedal robots [Bledt et al. 2018].

7.2 Automatic Motion Retargeting

We evaluate the performance of our pipeline for retargeting motion capture data to the Max family, Lizard, and ANYmal robots in simulation. We pair 13 frames on the robot with corresponding frames of the target motions and optimize the linear offset parameters. Additionally, we optimize non-uniform scaling of the reference motion parameterized by xy-scaling and z-scaling, resulting in a total of 41 open parameters in the upper-level optimization. We use one frame on the torso of the robot and define a *linear motion* tracking and an *angular motion* tracking task for it. For each leg, we define one frame per link with only *linear motion* tracking task. For both the *linear and angular motion* tasks we use diagonal weight matrices as $\mathbf{W_r} = \mathbf{W_A} = \mathrm{diag}[5, 5, 5]$ and $\mathbf{W_{\dot{r}}} = \mathbf{W_\omega} = \mathrm{diag}[1, 1, 1]$.

Initialization. We initialize the offset parameters to zero and the scaling coefficients by the relative base height in a canonical standing pose. The frame in the input animation that is paired with the torso of the robot is used to initialize the xy-position and yaw-orientation in the state trajectory, $\mathbf{x}(t)$. The remaining states and the input trajectory, $\mathbf{u}(t)$, are initialized with the canonical standing pose of the robot. After the first DOC iteration, each OC problem is warm-started with the solution of the previous iteration.

Generalization Across Robots. As seen in the accompanying video, our optimization can successfully retarget the dog's motion capture trajectories onto the different robots. One can observe differences in the retargeted motions for the extreme kinematics variations of Tall Max, Stocky Max, and Lizard. The same procedure is used to retarget the animation inputs (A1-A4) onto ANYmal, preparing for deployment on the physical robot. Selected frames of the retargeting of a jumping motion (M1) are shown in Fig. 6. During this motion, the dog makes great use of its flexible spine to transition between the different motion phases. Our simulated robots, having rigid spines, are able to compensate for the spine motion and find motions that maintain the qualities of the reference motions while respecting the robot dynamics and kinematics. The retargeting for Lizard is particularly pronounced because of its entirely different joint configuration.

Even when the kinematics of the robots are identical, significant difference can arise due to differences in dynamics. Fig. 7 shows how different contact forces arise due to the additional weight concentrated at the front of Pack Max.

Fig. 8 shows the optimized vertical scaling of the reference motion M8 for all robots in the Max Family. The reported physical height

is the torso height of each robot relative to max for the stance configuration shown in Fig. 5. The solver consistently converges to a solution close to the expected height.

Input Exceeding Robot Limits. We demonstrate the effect of actuator limits by comparing retargeting results for ANYmal and Strong ANYmal. As the two robots are identical apart from their actuator limits, the difference in the retargeted motion is solely due to these limits. For motion M6, with corresponding plots in Fig. 9, the weaker actuators often saturate during the motion, which results in significantly lower forward velocity and therefore jumping distance. Our setup accommodates the actuation limits by shrinking the motion in the xy-directions, thereby decreasing the actuation requirements. As best seen in the supporting video, the actuator limits are handled gracefully, preserving the motion, without introducing artefacts.

Non-physical Input. For motions M6 and M7, which start and end on a box respectively, our retargeting pipeline is able to retarget the motion to flat terrain. The jump in each of the motion is maintained while the foothold constraints in the optimal control problem enforce the solution to be physically consistent with the new environment.

7.3 Hardware Deployment

For the deployment on hardware, we reuse the MPC formulation described in Sec. 6.2. Instead of the retargeting objectives, the cost function from [Grandia et al. 2022] is used to track the already-retargeted state and input trajectories. Due to computational limits of the real system, the MPC horizon is fixed to 1.0 s and the controller is executed in a receding horizon fashion, re-optimizing the trajectory at 100 Hz. The output of the MPC layer is fed to a whole-body controller that uses a more accurate simulation model, but only solves for the next time step. At the lowest level is the joint control layer that commands the individual actuators.

Two retargeted motion clips are executed on the ANYmal hardware. The video shows the executed motion and retargeting result together. Because the retargeting is physically realistic, the system is able to closely follow the retargeted motion. Furthermore, through the use of online model-predictive control, the system is able to recover from disturbances and return to the motion reference when balance is regained (see video). Finally, we retarget four animation motion clips, as described in Table 1, to ANYmal and execute the motion clips on hardware. Here, unlike the motion capture data, there are no explicit physical constraints on the input motion. Frames comparing the creative animation and the end result on hardware are shown in Fig. 10.

7.4 Performance

Table 2 shows the required computation time of the retargeting pipeline for Max. Timings are reported for an Intel®Xeon®E-2178M CPU. Even though each iteration of the optimal control solver and each gradient computation scale linearly with the duration of the animation, the total solver time deviates from this. This is due to the variable number of iterations required at each level of the optimization.

Table 2. The computational time of the retargeting pipeline and number of iterations required by each component on a subset of the retarget motions.

Animation ID	M1	M2	M3	M4	M5	M6	M7	M8
Anim. Duration [s]	4.7	10.0	9.8	11.0	13.5	3.5	3.8	10.2
Solver Tot. Time [s]	129.7	138.6	139.2	148.3	265.8	75.7	57.5	177.7
DOCSens Time [s]	44.2	90.8	71.4	63.3	127.1	29.8	25.0	78.0
OC Time [s]	85.4	47.8	67.8	85.0	138.7	45.9	32.6	99.8
DOC iterations	10	13	13	12	12	15	9	11
DOCSens iterations	32	38	15	25	53	39	25	24
OC iterations	1555	419	616	655	856	1065	715	851

To highlight the performance of the upper-level optimization, we examine the convergence of the retargeting cost for Max in Fig. 11 (right). Here, we have normalized the retargeting cost with respect to the cost at the iteration zero which results from the initial parameters. Additionally, we show the cost convergence for the entire Max family for motion M8 in Fig. 11 (left). Qualitatively, the convergence behavior is independent of the robot and the motion.

The complexity of the MPC scales cubically with the number of state and control parameters.

7.5 Limitations

When a motion sequence contains full rotations of the robot, local minima arise due to the topology of rotations. Careful initialization of the torso trajectory, as described in Sec. 7.2, was found to work well for the motions used in this work. However, it does not guarantee that undesired minima are avoided. For acrobatic motions that include multiple rotations, a different strategy or global exploration might be required.

Self-collisions can occur due to differences in degrees of freedom and proportions between the input source and our robot family. An example can be found in the retargeted motion M5, see Fig. 12. The handling of self-collisions between the feet is left as future work.

8 CONCLUSION

We have devised a differential optimal control framework that interfaces with a large class of continuous optimal control problems with an additional set of equality constraints. By projecting the linearized equations onto the constraint manifold, we derive two similar sets of continuous matrix Riccati equations that allow us to compute a functional search direction to iteratively solve for optimal control trajectories as well as compute derivatives of optimal trajectories with respect to parameters.

Future Work. We assume the footfall pattern extracted from the input data to remain fixed throughout optimizations. This assumption may limit the quality of the retargeting result, especially for fast and dynamic input exceeding the limit of what the hardware can achieve. Letting the optimization make changes to this pattern, also referred to as optimal switching [Farshidian et al. 2017b; Li and Wensing 2020], could improve the retargeting result for challenging cases.

Our optimal retargeting is fast, but does not fulfill the requirements for a real-time deployment. The latter would enable real-time authoring of motions on hardware. Extensions of our algorithms that avoid dynamics derivatives [Plancher et al. 2017] or the exploration

Fig. 6. **Retargeting of mocap motion M1 onto Max and Lizard robots.** Top row shows the skeleton motion input, for representative frames. Middle row shows the result on Max; bottom row shows the result on Lizard. Despite the differences in kinematics between the input and the two robots, the characteristic traits of the motion are preserved. (Mirrored w.r.t. video.)

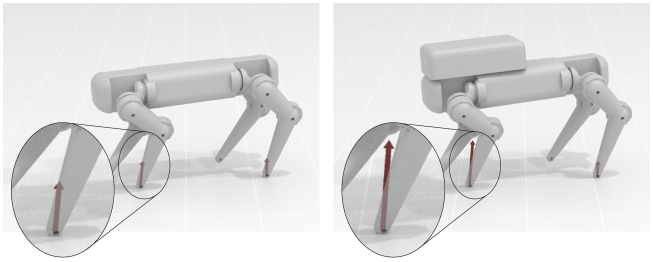

Fig. 7. Contact force visualization for Max (left) and Pack Max (right) at the same frame of motion M4. Due to the difference in mass distribution, the contact forces (inset) are significantly different, even for similar looking motions.

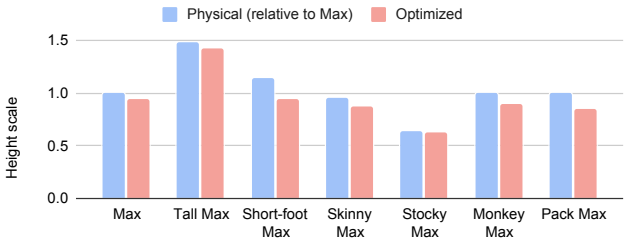

Fig. 8. Physical and optimized height scale for the Max Family on motion M8. The physical height scale is the torso height at a canonical stance relative to Max.

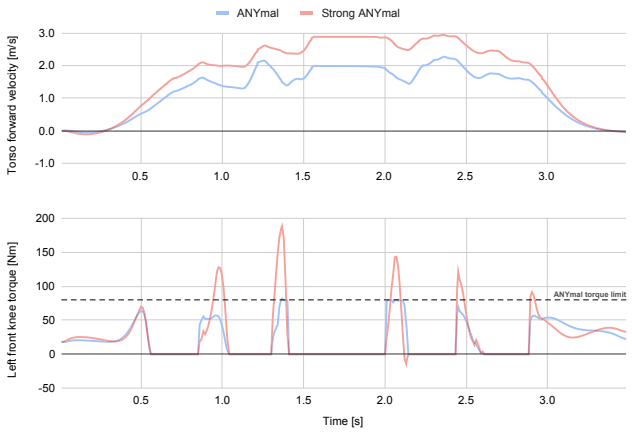

Fig. 9. **Effect of torque limits on retargeting result.** Plot of torso forward velocity and a representative motor torque for jumping motion (M6) retargeted onto ANYmal (torque limit 80 Nm) and Strong ANYmal (torque limit 200 Nm) robots. The optimization compensates for the limited torque of ANYmal by widening the torque spikes (bottom plot, e.g. at t=1.0 s), and also by scaling down the forward velocity of the motion (top plot). As seen in the supporting video, the limits are thus handled gracefully, while preserving the motion, without introducing artefacts.

of alternative simultaneous or sequential solution strategies [Hargraves and Paris 1987; Posa et al. 2016] are left as future work. Recent work on solving DDP problems under parameter uncertainty could improve robustness under sim-to-real gaps [Aoyama et al. 2021].

While we have used our differential optimal control to solve a retargeting problem for MPC, our method is agnostic to the underlying control problem and parameters. It is therefore applicable to generic equality-constrained optimal control problems, and has applications far beyond retargeting tasks we consider here.

ACKNOWLEDGMENTS

We wish to thank Violaine Fayolle and Doriano vanEssen for creating the animation that we retargeted onto ANYmal. This project

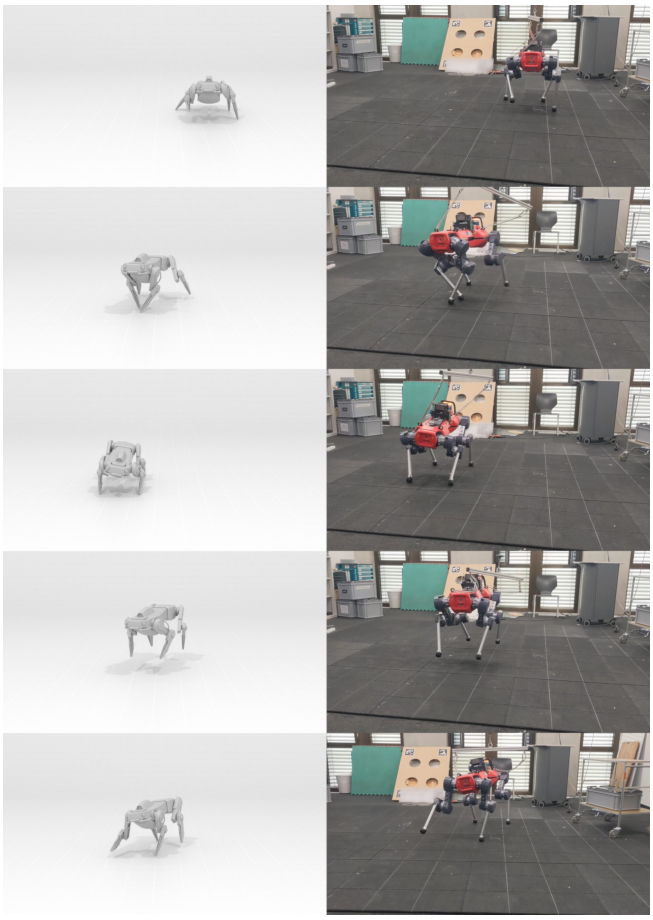

Fig. 10. **Retargeting of artist-designed animation onto ANYmal robot.** Selected frames from the input animation, alongside corresponding frames for the same motion executed on the physical robot. The robot motion remains visually close to the input, even though the original animation does not obey the laws of physics. See supporting video for full sequence.

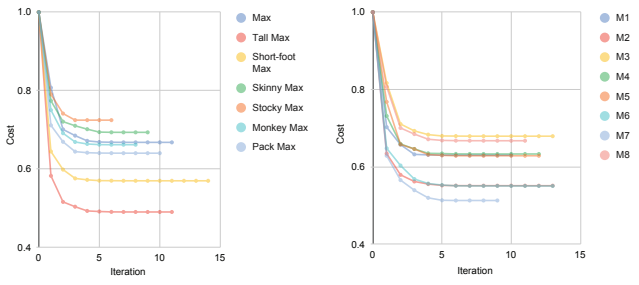

Fig. 11. Retargeting cost convergence for the Max Family on motion M8 (left). Retargeting cost convergence for Max on a selection of the retargeted motions (right). In both, the cost is normalized by the cost at initialization.

received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 852044, and was supported by the Swiss National Science Foundation (SNSF) as part of project No.188596.

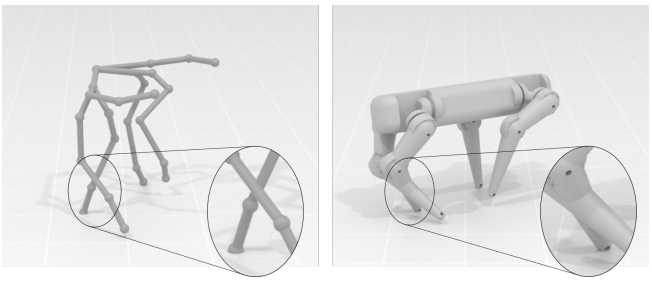

Fig. 12. Crossing of the hind legs in mocap motion M5 (left) leads to a self-collision when retargeting onto Max (right).

REFERENCES

Evan Ackerman. 2021. How Boston Dynamics Taught Its Robots to Dance. *IEEE Spectrum* (Jan. 7 2021). https://spectrum.ieee.org/how-boston-dynamics-taught-its-robots-to-dance

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. 2018. Differentiable MPC for end-to-end planning and control. *Advances in neural information processing systems* 31 (2018).

Yuichiro Aoyama, George Boutselis, Akash Patel, and Evangelos A. Theodorou. 2021. Constrained Differential Dynamic Programming Revisited. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

Taylor Apgar, Patrick Clary, Kevin Green, Alan Fern, and Jonathan Hurst. 2018. Fast Online Trajectory Optimization for the Bipedal Robot Cassie. In *Robotics: Science and Systems XIV*. Robotics: Science and Systems Foundation.

Ko Ayusawa and Eiichi Yoshida. 2017. Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization. *IEEE Transactions on Robotics* 33, 6 (2017), 1343–1357.

Dimitri Bertsekas. 2012. *Dynamic programming and optimal control: Volume I*. Vol. 1. Athena scientific.

Thomas Bi, Péter Fankhauser, Dario Bellicoso, and Marco Hutter. 2018. Real-Time Dance Generation to Music for a Legged Robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M. Wensing, and Sangbae Kim. 2018. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Steven Bohez, Saran Tunyasuvunakool, Philemon Brakel, Fereshteh Sadeghi, Leonard Hasenclever, Yuval Tassa, Emilio Parisotto, Jan Humplik, Tuomas Haarnoja, Roland Hafner, et al. 2022. Imitate and Repurpose: Learning Reusable Robot Movement Skills From Human and Animal Behaviors. *arXiv preprint arXiv:2203.17138* (2022).

SR McReynolds AE Bryson Jr. 1965. A successive sweep method for solving optimal programming problems. In *Proc 1965 Joint Automatic Control Conference*.

Sungjoon Choi, Matt Pan, and Joohyung Kim. 2020. Nonparametric motion retargeting for humanoid robots on shared latent space. In *16th Robotics: Science and Systems, RSS 2020*. MIT Press Journals.

Sungjoon Choi, Min Jae Song, Hyemin Ahn, and Joohyung Kim. 2021. Self-Supervised Motion Retargeting with Safety Guarantee. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 8097–8103.

Stelian Coros, Andrej Karpathy, Ben Jones, Lionel Reveret, and Michiel Van De Panne. 2011. Locomotion skills for simulated quadrupeds. *ACM Trans. Graph. (TOG)* 30, 4 (2011).

Marco Da Silva, Yeuhi Abe, and Jovan Popović. 2008. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library.

Behzad Dariush, Michael Gienger, Arjun Arumbakkam, Christian Goerick, Youding Zhu, and Kikuo Fujimura. 2008a. Online and markerless motion retargeting with kinematic constraints. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Behzad Dariush, Michael Gienger, Bing Jian, Christian Goerick, and Kikuo Fujimura. 2008b. Whole body humanoid control from human motion descriptors. In *2008 IEEE International Conference on Robotics and Automation*.

Kourosh Darvish, Yeshasvi Tirupachuri, Giulio Romualdi, Lorenzo Rapetti, Diego Ferigo, Francisco Javier Andrade Chavez, and Daniele Pucci. 2019. Whole-Body Geometric Retargeting for Humanoid Robots. In *2019 IEEE-RAS 19th International Conference*

on Humanoid Robots (Humanoids).

J.F.O. De O. Pantoja and D.Q. Mayne. 1989. A sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints. In *Proceedings of the 28th IEEE Conference on Decision and Control,*.

Farbod Farshidian, Edo Jelavic, Asutosh Satapathy, Markus Giftthaler, and Jonas Buchli. 2017a. Real-time motion planning of legged robots: A model predictive control approach. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*.

Farbod Farshidian, Michael Neunert, Alexander W Winkler, Gonzalo Rey, and Jonas Buchli. 2017b. An efficient optimal planning and control framework for quadrupedal locomotion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Christian Gehring, Stelian Coros, Marco Hutter, Carmine Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Peter Fankhauser, Jemin Hwangbo, Mark Hoepflinger, and Roland Siegwart. 2016. Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot. *IEEE Robotics & Automation Magazine* 23, 1 (2016).

Michael J. Gielniak, C. Karen Liu, and Andrea L. Thomaz. 2010. Secondary action in robot motion. In *19th International Symposium in Robot and Human Interactive Communication*. IEEE, Viareggio, Italy.

Michael J. Gielniak and Andrea L. Thomaz. 2011. Spatiotemporal correspondence as a metric for human-like robot motion. In *2011 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*.

Markus Giftthaler and Jonas Buchli. 2017. A projection approach to equality constrained iterative linear quadratic optimal control. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*.

Michael Gleicher. 1998. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. ACM Press.

Ruben Grandia, Farbod Farshidian, René Ranftl, and Marco Hutter. 2019. Feedback MPC for torque-controlled legged robots. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. 2022. Perceptive Locomotion through Nonlinear Model Predictive Control. *https://arxiv.org/pdf/2208.08373.pdf* (2022).

C.R. Hargraves and S.W. Paris. 1987. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics* 10, 4 (1987).

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. 2018. Learning an Embedding Space for Transferable Robot Skills. https://openreview.net/forum?id=rk07ZXZRb

Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. 2008. Real-time motion retargeting to highly varied user-created morphologies. *ACM Trans. Graph. (TOG)* 27, 3 (2008).

Shayan Hoshyari, Hongyi Xu, Espen Knoop, Stelian Coros, and Moritz Bächer. 2019. Vibration-minimizing motion retargeting for robotic characters. *ACM Trans. Graph. (TOG)* 38, 4 (2019).

Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. 2019. ALTRO: A Fast Solver for Constrained Trajectory Optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

David H. Jacobson and David Q. Mayne. 1970. *Differential Dynamic Programming*. American Elsevier Publishing Company.

Dongho Kang, Flavio De Vincenti, Naomi C. Adami, and Stelian Coros. 2022. Animal Motions on Legged Robots Using Nonlinear Model Predictive Control. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Dongho Kang, Simon Zimmermann, and Stelian Coros. 2021. Animal Gaits on Quadrupedal Robots Using Motion Matching and Model-Based Control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Benjamin Katz, Jared Di Carlo, and Sangbae Kim. 2019. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE.

Sunwoo Kim, Maks Sorokin, Jehee Lee, and Sehoon Ha. 2022. HumanConQuad: Human Motion Control of Quadrupedal Robots using Deep Reinforcement Learning. In *SIGGRAPH Asia 2022 Emerging Technologies*. ACM.

Seyoung Lee, Sunmin Lee, Yongwoo Lee, and Jehee Lee. 2021. Learning a family of motor skills from a single motion clip. *ACM Transactions on Graphics* 40, 4 (2021).

He Li and Patrick M. Wensing. 2020. Hybrid Systems Differential Dynamic Programming for Whole-Body Motion Planning of Legged Robots. *IEEE Robotics and Automation Letters* 5, 4 (2020).

Tianyu Li, Jungdam Won, Sehoon Ha, and Akshara Rai. 2021. Model-based Motion Imitation for Agile, Diverse and Generalizable Quadupedal Locomotion. *arXiv preprint arXiv:2109.13362* (2021).

Weiwei Li and Emanuel Todorov. 2004. Iterative linear quadratic regulator design for nonlinear biological movement systems.. In *ICINCO (1)*.

T. C. Lin and J. S. Arora. 1991. Differential dynamic programming technique for constrained optimal control. *Computational Mechanics* 9, 1 (1991).

David Mayne. 1966. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *Internat. J. Control* 3, 1 (1966).

S. K. Mitter. 1966. Successive approximation methods for the solution of optimal control problems. *Automatica* 3, 3 (1966).

Daniel M. Murray and Sidney J. Yakowitz. 1979. Constrained differential dynamic programming and its application to multireservoir control. *Water Resources Research* 15, 5 (1979).

Shin'ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa, and Katsushi Ikeuchi. 2007. Learning from Observation Paradigm: Leg Task Models for Enabling a Biped Humanoid Robot to Imitate Human Dances. *The International Journal of Robotics Research* 26, 8 (2007).

Michael Neunert, Cédric de Crousaz, Fadri Furrer, Mina Kamel, Farbod Farshidian, Roland Siegwart, and Jonas Buchli. 2016. Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

Michael Neunert, Farbod Farshidian, Alexander W. Winkler, and Jonas Buchli. 2017. Trajectory Optimization Through Contacts and Automatic Gait Discovery for Quadrupeds. *IEEE Robotics and Automation Letters* 2, 3 (2017).

Koichi Nishiwaki, Satoshi Kagami, Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. 2002. Online generation of humanoid walking motion based on a fast generation method of motion pattern that follows desired zmp. In *IEEE/RSJ international conference on intelligent robots and systems*, Vol. 3. IEEE, 2684–2689.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization*. Springer, New York, NY, USA.

Alex Oshin, Matthew D. Houghton, Michael J. Acheson, Irene M. Gregory, and Evangelos A. Theodorou. 2022. Parameterized Differential Dynamic Programming.

Christian Ott, Dongheui Lee, and Yoshihiko Nakamura. 2008. Motion capture based human motion recognition and imitation by direct marker control. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*.

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. 2020. Learning agile robotic locomotion skills by imitating animals. *Robotics: Science and Systems* (2020).

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. on Graph. (TOG)* 41, 4 (2022).

Brian Plancher, Zachary Manchester, and Scott Kuindersma. 2017. Constrained unscented dynamic programming. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

N.S. Pollard, J.K. Hodgins, M.J. Riley, and C.G. Atkeson. 2002. Adapting human motion for the control of a humanoid robot. In *Proceedings 2002 IEEE International Conference on Robotics and Automation*, Vol. 2. IEEE, Washington, DC, USA.

Michael Posa, Scott Kuindersma, and Russ Tedrake. 2016. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*.

Marc H Raibert and Jessica K Hodgins. 1991. Animation of dynamic legged locomotion. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 349–358.

Lionel Reveret, Laurent Favreau, Christine Depraz, and Marie-Paule Cani. 2005. Morphable model of quadrupeds skeletons for animating 3d animals. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 135–142.

Quentin Rouxel, Kai Yuan, Ruoshi Wen, and Zhibin Li. 2022. Multicontact Motion Retargeting Using Whole-Body Optimization of Full Kinematics and Sequential Force Equilibrium. *IEEE/ASME Transactions on Mechatronics* (2022).

Hoseok Ryu, Minseok Kim, Seungwhan Lee, Moon Seok Park, Kyoungmin Lee, and Jehee Lee. 2021. Functionality-Driven Musculature Retargeting. *Computer Graphics Forum* 40, 1 (2021).

Christian Schumacher, Espen Knoop, and Moritz Bächer. 2021. A Versatile Inverse Kinematics Formulation for Retargeting Motions onto Robots with Kinematic Loops. (2021).

Jian Shi, Peter B Luh, Shi-Chung Chang, and Tsu-Shuan Chang. 1990. A Method for Constrained Dynamic Optimization Problems. In *1990 American Control Conference*.

Ljiljana Skrba, Lionel Reveret, Franck Hétroy, Marie-Paule Cani, and Carol O'Sullivan. 2009. Animating quadrupeds: methods and applications. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library.

Jean-Pierre Sleiman, Farbod Farshidian, and Marco Hutter. 2021. Constraint Handling in Continuous-Time DDP-Based Model Predictive Control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

Laura Smith, J Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. 2021. Legged Robots that Keep on Learning: Fine-Tuning Locomotion Policies in the Real World. *arXiv preprint arXiv:2110.05457* (2021).

Kwang Won Sok, Manmyung Kim, and Jehee Lee. 2007. Simulating biped behaviors from human motion data. *ACM Trans. Graph.* 26, 3 (2007).

Wei Sun, Evangelos A. Theodorou, and Panagiotis Tsiotras. 2014. Continuous-time differential dynamic programming with terminal constraints. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. 2018. Sim-to-real: Learning agile locomotion for quadruped robots. *Robotics: Science and Systems* (2018).

Yuval Tassa, Tom Erez, and Emanuel Todorov. 2012. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE.

Yuval Tassa, Nicolas Mansard, and Emo Todorov. 2014. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

Michael Taylor, Sergey Bashkirov, Javier Fernandez Rico, Ike Toriyama, Naoyuki Miyada, Hideki Yanagisawa, and Kensaku Ishizuka. 2021. Learning Bipedal Robot Locomotion from Human Movement. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2797–2803.

E. Todorov and Weiwei Li. 2005. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference, 2005.*

Tarik Tosun, Ross Mead, and Robert Stengel. 2014. A general method for kinematic retargeting: Adapting poses between humans and robots. In *ASME International Mechanical Engineering Congress and Exposition*. American Society of Mechanical Engineers.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics* 41, 4 (2022).

Jungdam Won and Jehee Lee. 2019. Learning body shape variation in physics-based characters. *ACM Trans. Graph. (TOG)* 38, 6 (2019), 1–12.

Weitao Xi and C. David Remy. 2014. Optimal gaits and motions for legged robots. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Zhaoming Xie, C. Karen Liu, and Kris Hauser. 2017. Differential dynamic programming with nonlinear constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*.

Katsu Yamane, Stuart O Anderson, and Jessica K Hodgins. 2010. Controlling humanoid robots with human motion data: Experimental validation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*. IEEE.

Chuanyu Yang, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li. 2020. Multi-expert learning of adaptive legged locomotion. *Science Robotics* 5, 49 (2020), eabb2174.

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022a. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Transactions on Graphics* 41, 6 (2022).

Qingfeng Yao, Jilong Wang, Shuyu Yang, Cong Wang, Hongyin Zhang, Qifeng Zhang, and Donglin Wang. 2022b. Imitation and Adaptation Based on Consistency: A Quadruped Robot Imitates Animals from Videos Using Deep Reinforcement Learning. *arXiv preprint arXiv:2203.05973* (2022).

Yuting Ye and C. Karen Liu. 2010. Optimal feedback control for character animation using an abstract model. *ACM Transactions on Graphics* 29, 4 (2010).

Fulong Yin, Annan Tang, Liangwei Xu, Yue Cao, Yu Zheng, Zhengyou Zhang, and Xiangyu Chen. 2021. Run Like a Dog: Learning Based Whole-Body Control Framework for Quadruped Gait Style Transfer. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph. (TOG)* 37, 4 (2018), 1–11.

Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. 2020. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Trans. on Graph. (TOG)* 39, 6 (2020).

Ziyi Zhou, Bruce Wingo, Nathan Boyd, Seth Hutchinson, and Ye Zhao. 2022. Momentum-Aware Trajectory Optimization and Control for Agile Quadrupedal Locomotion. *arXiv preprint arXiv:2203.01548* (2022).

Victor Brian Zordan and Jessica K. Hodgins. 2002. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02)*. Association for Computing Machinery, New York, NY, USA.

APPENDIX

In Sec. 5, we reduced a generic linear two-boundary problem with four equations and variables to a problem with two equations and unknowns. We provide expressions for the matrix coefficients of the 3x3 matrix $\tilde{\boldsymbol{\mathcal{H}}}$

$$\begin{aligned} \tilde{H}_{\mathbf{xx}} &= H_{\mathbf{xx}} + H_{\mathbf{xu}}\mathbf{P}_{\mathbf{x}} + \mathbf{P}_{\mathbf{x}}^T H_{\mathbf{ux}} + \mathbf{P}_{\mathbf{x}}^T \mathbf{H}_{\mathbf{uu}}\mathbf{P}_{\mathbf{x}} & \tilde{H}_{\mathbf{uu}} &= \mathbf{P}_{\mathbf{u}}^T H_{\mathbf{uu}}\mathbf{P}_{\mathbf{u}} \\ \tilde{H}_{\mathbf{ux}} &= \mathbf{P}_{\mathbf{u}}^T H_{\mathbf{ux}} + \mathbf{P}_{\mathbf{u}}^T H_{\mathbf{uu}}\mathbf{P}_{\mathbf{x}} & \tilde{H}_{\mathbf{xu}} &= \tilde{H}_{\mathbf{ux}}^T \\ \tilde{\mathbf{f}}_{\mathbf{x}} &= \mathbf{f}_{\mathbf{x}} + \mathbf{f}_{\mathbf{u}}\mathbf{P}_{\mathbf{x}} & \tilde{\mathbf{f}}_{\mathbf{u}} &= \mathbf{f}_{\mathbf{u}}\mathbf{P}_{\mathbf{u}} \end{aligned} \tag{32}$$

and the corresponding constant vector $\tilde{\boldsymbol{C}}$

$$\tilde{H}_{\mathbf{x}} = H_{\mathbf{x}} + H_{\mathbf{xu}}\mathbf{P} + \mathbf{P}_{\mathbf{x}}^T H_{\mathbf{u}} + \mathbf{P}_{\mathbf{x}}^T H_{\mathbf{uu}}\mathbf{P} \quad \tilde{H}_{\mathbf{u}} = \mathbf{P}_{\mathbf{u}}^T H_{\mathbf{u}} + \mathbf{P}_{\mathbf{u}}^T H_{\mathbf{uu}}\mathbf{P} \tag{33}$$
$$\tilde{\mathbf{f}} = \mathbf{f} + \mathbf{f}_{\mathbf{u}}\mathbf{P}$$

in the system 21, and for the coefficients of $\hat{\boldsymbol{\mathcal{H}}}$

$$\hat{H}_{\mathbf{xx}} = \tilde{H}_{\mathbf{xu}}\tilde{H}_{\mathbf{uu}}^{-1}\tilde{H}_{\mathbf{ux}} - \tilde{H}_{\mathbf{xx}} \qquad \hat{\mathbf{H}} = -\tilde{\mathbf{f}}_{\mathbf{u}}\tilde{H}_{\mathbf{uu}}^{-1}\tilde{\mathbf{f}}_{\mathbf{u}}^T \tag{34}$$
$$\hat{\mathbf{f}}_{\mathbf{x}} = \tilde{\mathbf{f}}_{\mathbf{x}} - \tilde{\mathbf{f}}_{\mathbf{u}}\tilde{H}_{\mathbf{uu}}^{-1}\tilde{H}_{\mathbf{ux}}$$

and vector coefficients $\hat{\boldsymbol{C}}$

$$\hat{H}_{\mathbf{x}} = \tilde{H}_{\mathbf{xu}}\tilde{H}_{\mathbf{uu}}^{-1}\tilde{H}_{\mathbf{u}} - \tilde{H}_{\mathbf{x}} \qquad \hat{\mathbf{f}} = \tilde{\mathbf{f}} - \tilde{\mathbf{f}}_{\mathbf{u}}\tilde{H}_{\mathbf{uu}}^{-1}\tilde{H}_{\mathbf{u}} \tag{35}$$

in the 2x2 problem 23.