Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Multi-Phase Optimal Control Problems for Efficient Nonlinear Model Predictive Control with acados

Jonathan Frey[1,2] | **Katrin Baumgärtner**[1] | **Gianluca Frison**[1] | **Moritz Diehl**[1,2]

[1]Department of Microsystems Engineering (IMTEK), University Freiburg, Germany

[2]Department of Mathematics, University Freiburg, Germany

Correspondence
Corresponding author Jonathan Frey.
Systems Control and Optimization Laboratory, Department of Microsystems Engineering (IMTEK), University of Freiburg, Freiburg im Breisgau, Germany
Email: jonathan.frey@imtek.uni-freiburg.de

Abstract

Computationally efficient nonlinear model predictive control relies on elaborate discrete-time optimal control problem (OCP) formulations trading off accuracy with respect to the continuous-time problem and associated computational burden. Such formulations, however, are in general not easy to implement within specialized software frameworks tailored to numerical optimal control. This paper introduces a new multi-phase OCP interface for the open-source software `acados` allowing to conveniently formulate such problems and generate fast solvers that can be used for nonlinear model predictive control (NMPC). While multi-phase OCP (MOCP) formulations occur naturally in many applications, this work focuses on MOCP formulations that can be used to efficiently approximate standard continuous-time OCPs in the context of NMPC. To this end, the paper discusses advanced control parametrizations, such as closed-loop costing and piecewise polynomials with varying degree, as well as partial tightening and formulations that leverage models of different fidelity. An introductory example is presented to showcase the usability of the new interface. Finally, three numerical experiments demonstrate that NMPC controllers based on multi-phase formulations can efficiently trade-off computation time and control performance.

KEYWORDS
nonlinear model predictive control, real-time control, optimal control,

1 | INTRODUCTION

High-performance algorithms for nonlinear model predictive control (NMPC) have extended its real-time applicability from chemical processes to a huge variety of application areas [1,2,3,4,5,6,7,8]. Most NMPC implementations use a discrete-time model with a fixed time step that is equal to the sampling time. However, more elaborate optimal control problem (OCP) formulations have enormous potential to reduce the computational burden associated with the online optimization. Nonuniform time grids, which have been shown to work especially well with accurate cost integration [9], are an easy first step in this direction. However, more sophisticated OCP formulations, like piecewise polynomial control parametrizations of varying degree [10], closed-loop-costing formulations [11], or using models of different fidelity for parts of the horizon, are rarely used in practice, due to the additional implementation effort. Such formulations require the problem functions as well as input and state dimensions to vary over the horizon. To this end, this paper presents a new feature of the open-source software `acados` [12] which allows for a convenient formulation of multi-phase OCPs via its `Python` interface.

We use the term *multi-phase* OCP (MOCP) to describe OCP formulations that may have structurally different models, constraints and cost formulations on parts of the horizon. In contrast the term *multi-stage* was used in previous works but is used in different ways, e.g. in the context of tree-structured OCPs [13,14,15] and to describe OCP formulations which only allow to vary cost and constraint functions [16]. The proposed multi-phase formulation is similar to the one tackled by `GPOPS-II` [17], which is a commercial `Matlab` software package for solving multi-phase OCPs using state-of-the-art sparse NLP solvers, like `IPOPT` [18] and `SNOPT` [19]. `GPOPS-II` has been successfully used for MOCPs [20].

However, the focus of this paper is on MOCP formulations with OCP-structure exploiting algorithms. MOCPs have been previously tackled in the direct multiple shooting package MUSCOD-II [21], which allows multi-phase formulations with piecewise

polynomial control parametrizations and coupled processes. While MUSCOD-II focused on problems from process control with time scales in the second to minute range,[21,22], the acados software package has been successfully deployed on systems with much shorter time scales[2,3,4,5,6,7,8]. Both MUSCOD-II and acados use a direct multiple shooting based formulation.

The open-source software package acados implements efficient algorithms for embedded optimal control[12]. It is written in C and relies on the linear algebra package BLASFEO which provides performance-optimized routines for small to medium sized matrix operations[23]. The nonlinear OCPs can be solved with acados using variants of sequential quadratic programming (SQP), the real-time iteration (RTI)[24] and advanced-step real-time iteration (AS-RTI) scheme[25], as well as differential dynamic programming (DDP)[26], which was recently added[27]. In acados, the quadratic subproblems are solved exploiting the block structure of optimal control problems. It has been shown that OCP structure exploiting solvers can be many times faster compared to general sparse solvers or dense ones used with condensing[28,29]. A variety of quadratic programming (QP) solvers, such as HPIPM, qpOASES, DAQP, OSQP, qpDUNES,[30,31,32,33,34] are interfaced, which either tackle the OCP-structured QP directly or after applying full or partial condensing to it[35,36]. The system dynamics can be handled with integration methods which are able to propagate forward and adjoint sensitivities efficiently in addition to the nominal result[12,37].

The remainder of this introduction aims at giving a brief overview on software frameworks for NMPC other than acados. The tool CasADi[38] offers a convenient symbolic framework, algorithmic differentiation and interfaces to many state-the-art NLP solvers, like IPOPT[18] and SNOPT[19]. A variety of software projects specialized on OCP formulations have been developed on top of CasADis symbolic framework. The package do-mpc[14], provides Python functionality to allow fast prototyping of NMPC, moving horizon estimation (MHE) and supports tree-structured OCP formulations. In terms of solvers it relies on the ones available in CasADi. The rockit package[39], allows to conveniently formulate OCPs in Python and Matlab and also covers MOCP formulations. In addition to the CasADi solvers, it also provides interfaces to acados and fatrop[29], which is an NLP solver highly inspired by IPOPT that exploits the optimal control problem structure. The OpEn[40] software package allows to conveniently formulate single-phase OCP problems and implements the proximal averaged Newton-type method for optimal control (PANOC) as a Rust solver and allows to generate solvers for user defined problems from Python and Matlab using the CasADi symbolics. The GRAMPC[41] software package can generate embedded solvers from Matlab and uses a gradient-based augmented Lagrangian method. The commercial solver FORCESPRO[42] implements competitive algorithms for NMPC which support varying dimensions between the stages[42,43]. However, benchmark results created with the academic license of FORCESPRO could not be disclosed in past research[28,7].

Outline. The paper introduces the multi-phase OCP formulation in Section 2 and discusses how it can be handled using direct multiple shooting. Section 3 motivates using advanced OCP formulations which may be cast as MOCPs to design efficient NMPC controllers. In particular, it discusses piecewise polynomial control parameterizations, partial and progressive tightening formulations, and the use of models of different fidelity within a single OCP. Moreover, it presents and conceptually extends closed-loop costing formulations. Section 4 discusses the efficient treatment of multi-phase formulations within the acados software package and presents a tutorial example. Section 5 demonstrates how NMPC controllers based on the MOCP formulations detailed in Section 3 are able to trade off computation time and control performance in ways that are not accessible when one is limited to single-phase OCPs.

2 | MULTI-PHASE OPTIMAL CONTROL PROBLEM FORMULATIONS

This section presents a continuous-time multi-phase optimal control problem (MOCP) formulation in Section 2.1 and describes how it can be discretized using direct multiple shooting in Section 2.2.

2.1 | Continuous-time multi-phase optimal control problem

Throughout this paper, we will treat continuous-time multi-phase optimal control problems (MOCP), which can be stated as

$$\begin{aligned}
\underset{\substack{\xi_1(\cdot),\dots,\xi_M(\cdot),\\ v_1(\cdot),\dots,v_M(\cdot),\\ \eta_1,\dots,\eta_M}}{\text{minimize}} \quad & \sum_{k=1}^{M} \int_{t_k}^{t_{k+1}} \ell_k(\xi_k(t), v_k(t))\,\mathrm{d}t + E_k(\xi_k(t_{k+1}), \eta_k) && \text{(1a)}\\
\text{subject to} \quad & 0 = \bar{x}_0 - \xi_1(t_1), && \text{(1b)}\\
& 0 = f_k(t, \xi_k(t), \dot{\xi}_k(t), v_k(t)), && \text{(1c)}\\
& 0 \geq g_k(\xi_k(t), v_k(t)), && \text{(1d)}\\
& 0 = \Gamma_k(\xi_k(t_{k+1}), \eta_k) - \xi_{k+1}(t_{k+1}), && \text{(1e)}\\
& 0 \geq g_\mathrm{e}(\xi_M(T)), && \text{(1f)}\\
& \qquad \text{for } t \in [t_k, t_{k+1}),\ k = 1, \dots, M.
\end{aligned}$$

The finite time horizon $[0, T]$ is split into M fixed subintervals $[t_k, t_{k+1}]$ with $t_1 = 0$ and $t_{M+1} = T$. Each interval defines a phase $k \in \{1, \dots, M\}$. For each phase k, an implicit ODE f_k is given, which defines the state trajectory $\xi_k : [t_k, t_{k+1}] \to \mathbb{R}^{n_{x,k}}$ for a given control trajectory $v_k : [t_k, t_{k+1}] \to \mathbb{R}^{n_{v,k}}$ and initial state. The initial state of the first phase is given by $\bar{x}_0$, while the initial state for the subsequent phases is given by the transition functions $\Gamma_k : \mathbb{R}^{n_{x,k}} \times \mathbb{R}^{n_{\eta,k}} \to \mathbb{R}^{n_{x,k+1}}$, which map the terminal state of phase k and discrete decision variables $\eta_k \in \mathbb{R}^{n_{\eta,k}}$ to the initial state of the next phase, allowing for the state dimension to vary between the phases. The functions ℓ_k and E_k define the path and terminal cost terms for phase k. The functions g_k summarize the inequalities imposed on states and controls in phase k. Additionally, the function g_e summarizes the terminal constraints imposed at the end of the horizon.

The transition formulation (1e) is similar to the one in MUSCOD-II[44] with the addition of the discrete decision variable η_k. Additionally, the formulation in the work by Leineweber et al.[44] considers global variables and the time grid points t_i as optimization variables. The global variables are implemented as separate variables on each shooting interval and are constrained to be equal. Global variables can be formulated in (1) by state augmentation on the full horizon. In order to have time grid points as optimization variables, the dynamics can be augmented with a clock state and a speed of time variable acting as a control. The differences in problem formulation can be motivated by the different solution methods implemented in `acados` and MUSCOD-II, respectively. While MUSCOD-II deploys sparse linear algebra on blocks and allows linear couplings between stages, `acados` uses specialized algorithms for purely OCP-structured problems in which the dynamics constraints are the only coupling between stages.

MOCP formulations occur naturally in different application when formulating OCPs where the dynamic behavior qualitatively changes at a certain point in time. For example, when considering walking robots[45], chemical plants, where some amount of substance is added at a certain point in time, e.g. when considering recycled waste cuts[46], or multi-train scheduling[20]. However, the main focus of this paper is on MOCP formulations which can be derived to approximate the solution of a continuous-time infinite-horizon problem in the context of NMPC as discussed next.

2.2 | Multi-phase multiple shooting discretization

One can discretize MOCP (1) using direct multiple shooting[47] dividing each time interval $[t_k, t_{k+1}]$ into N_k shooting intervals. The resulting nonlinear program (NLP) can be written as

$$\underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{\eta}}{\text{minimize}} \quad \sum_{k=1}^{M}\sum_{j=0}^{N_k-1} L_{k,j}(x_{k,j}, u_{k,j}, \Delta t_{k,j}) + E_k(x_{k,N_k}, \eta_k) \tag{2a}$$

$$\text{subject to} \quad x_{1,0} = \bar{x}_0, \tag{2b}$$

$$x_{k+1,0} = \Gamma_k(x_{k,N_k}, \eta_k), \quad k = 1, \dots M-1, \tag{2c}$$

$$x_{k,j+1} = \phi_{k,j}(x_{k,j}, u_{k,j}),\ j = 0, \dots, N_k - 1, k = 1, \dots M, \tag{2d}$$

$$0 \geq g_{k,j}(x_{k,j}, u_{k,j}),\ j = 0, \dots, N_k - 1, k = 1, \dots M, \tag{2e}$$

$$0 \geq g_{\mathrm{e}}(x_{M,N_M}), \tag{2f}$$

where $\boldsymbol{x} = (x_{1,0}, \dots, x_{1,N_1}, x_{2,0}, \dots, x_{M,N_M})$, $\boldsymbol{u} = (u_{1,0}, \dots, u_{1,N_1-1}, u_{2,0}, \dots, u_{M,N_M-1})$, $\boldsymbol{\eta} = (\eta_1, \dots, \eta_M)$. The discrete values $x_{k,j} \in \mathbb{R}^{n_{x,k}}$ represent the values of $\xi_k(\cdot)$ at the corresponding shooting nodes. The controls $u_{k,j} \in \mathbb{R}^{n_{u,k}}$ act on the jth shooting interval of phase k. The cost functions $L_{k,j}$ are called stage costs and reflect the cost integrated over the shooting interval $[t_{k,j}, t_{k,j+1})$ with $\Delta t_{k,j} = t_{k,j+1} - t_{k,j}$. The initial state of the first phase is given by (2b) and for subsequent stages by (2c). The discrete-time dynamics are described by the functions $\phi_{k,j}$, which represent an integration scheme applied to the continuous-time dynamic system in (1c). Dedicated functions that compute the numerical integration of continuous-time dynamics and the solution sensitivities are an essential component of for efficient solution of OCPs and available in `acados`[37].

Finally, the constraint functions $g_{k,j}$ represent the constraints g_k on shooting interval $[t_{k,j}, t_{k,j+1})$. Most commonly, the constraints are only enforced at the initial point of the shooting interval, i.e. for $x_{k,j}, u_{k,j}$. However, it is possible to also impose them on intermediate points. Similarly, the stage cost $L_{k,j}$ often corresponds to a simple Euler integration of the continuous-time cost ℓ_k over a shooting interval. Especially, when using longer intervals, higher order integration of the cost term are necessary to retain a good approximation quality. The cost integration can be performed efficiently together with the integration of the dynamics (1c). More details on this can be found in Section 3.4.1.

3 | MULTI-PHASE OCP FORMULATIONS FOR MODEL PREDICTIVE CONTROL

This section motivates the use of multi-phase optimal control problem (MOCP) formulations for NMPC applications, in which an ideal controller would apply the exact solution to a continuous-time infinite-horizon optimal control problem, which is presented in Section 3.1. Moreover, Section 3.1 introduces the concept of the cost-to-go function, the importance of its approximation in NMPC, and why MOCP formulations should be considered in this regard.

The following subsections provide examples of optimal control problem formulations which are suitable to approximate parts of the cost-to-go term corresponding to different parts of the time horizon in successively more approximate ways. These formulations can be cast as multi-phase problems and can thus be solved efficiently using `acados`. In particular, Section 3.2 motivates using models of different fideltiy for different parts of the horizon, Section 3.3 discusses piecewise polynomial control parameterizations, Section 3.4 discusses closed-loop costing and Section 3.5 presents partial tightening.

3.1 | Continuous-time optimal control problem, model predictive control and cost-to-go

The standard infinite-horizon continuous-time optimal control problem (OCP) which we aim at approximating in various ways in this paper can be written as

$$
\begin{aligned}
V(\bar{x}_0) = \min_{x(\cdot),v(\cdot)} \quad & \int_0^\infty \ell(x(t), v(t))\,\mathrm{d}t && \text{(3a)}\\
\text{s.t.} \quad & \bar{x}_0 = x(0), && \text{(3b)}\\
& 0 = f(x(t), \dot{x}(t), v(t)), \quad t \in [0, \infty), && \text{(3c)}\\
& 0 \geq g(x(t), v(t)), \qquad t \in [0, \infty), && \text{(3d)}
\end{aligned}
$$

where $x : [0, \infty) \to \mathbb{R}^{n_x}$, $v : [0, \infty) \to \mathbb{R}^{n_v}$ are the state and control trajectories respectively, $\bar{x}_0$ is the initial state value, the function f describes the implicit system dynamics and g denotes the inequality constraints. The optimal value of OCP (3) is defined as $V(\bar{x}_0)$. The function $V(\cdot)$ is called cost-to-go and corresponds to the value function in the field of reinforcement learning.

The goal of model predictive control (MPC) is to operate a system, which typically applies a constant control input for a fixed sampling time Δt. This practical constraint can be formalized as

$$
v(t) = u_0 \quad \text{for } t \in [0, \Delta t]. \tag{4}
$$

Regarding OCP (3) from a dynamic programming point of view allows us to split the infinite horizon in different parts which might be approximated in different ways. Using the principle of optimality and accounting for (4), we can rewrite (3) as

$$
\begin{aligned}
\underset{x(\cdot),u_0}{\text{minimize}} \quad & \int_0^{\Delta t} \ell(x(t), u_0)\,\mathrm{d}t + V(x(\Delta t))\\
\text{subject to} \quad & \bar{x}_0 = x(0),\\
& 0 = f(x(t), \dot{x}(t), u_0), \; t \in [0, \Delta t],\\
& 0 \geq g(x(t), v(t)), \quad t \in [0, \Delta t),
\end{aligned} \tag{5}
$$

which is again an OCP but with a much shorter horizon of length Δt. Of course, the whole complexity of the problem is shifted into the minimization of the cost-to-go term with this reformulation. However, this shows that approximating the cost-to-go is key in the developement of an efficient NMPC controller.

In order to obtain an OCP formulation for MPC, one typically selects a finite time horizon, replaces the infinite integral in (3) with a finite one and adds a terminal cost on the terminal state. This approximation of the cost-to-go together with (5) can be discretized using multiple shooting, where the first shooting interval typically corresponds to (5).

Additionally, the control trajectory to approximate the closed-loop cost could be parameterized in different ways. In particular, piecewise polynomial control parameterizations and closed-loop costing are dicussed in Section 3.3 and Section 3.4. These parameterizations correspond to piecewise continuous functions, that are not applicable to the real system, given the practical constraint (4). However, since the cost-to-go is anyway approximated, such control parametrizations can lead to better approximations of the cost-to-go when using longer intervals.

In the following, we derive and discuss various multiple shooting based approximate versions of (3) which apply structurally different approximations for parts of the infinite horizon in (3) and can be be phrased as MOCPs.

3.2 | Models of different fidelity

In many physical systems, models of different fidelity are available and choosing an appropriate one for NMPC might depend on the control frequency, desired time horizon and the available solvers. The consideration of high-fidelity models is typically beneficial for MPC performance. However, such models might be computationally demanding and can become unstable for long time horizons, which are important to consider in OCPs in order to capture the evolution of slow dynamics and economic cost terms. Multi-phase OCP formulations allow one to choose different models for specific parts of the horizon, allowing for many more degrees of freedom when deriving a discrete-time finite horizon problem. Low-fidelty models are typically cheaper to

integrate with a certain accuracy, not only due to their potentially reduced dimensionality, but also because they are usually less stiff and thus can be handled by computationally cheaper integration schemes, such as explicit Runge-Kutta (ERK) methods. On the other hand, high fidelty models typically comprise both fast and slow modes, which causes them to be stiff. Such stiff models need to be handled with computationally expensive implicit Runge-Kutta (IRK) methods or a great number of very small integration steps of ERK methods.

Moreover, some nonlinear constraints, which are expensive to evaluate, could be only included in a first part of the horizon. This can be motivated by the practical observation that constraints tend to be active in the first part of the horizon[48]. For example, modelling oscillations within a physical system can be expensive and only meaningful on short time horizons, as in the application of wind turbine control[49].

Lastly, a variety of NMPC applications rely on underlying controllers that handle the actuators. However, it can be beneficial to directly model these actuators to accurately represent their behavior at least in the first part of the horizon. This can allow one to specify cost and constraints that need a model of the underlying actuator. Including such considerations can unleash optimality potential that is inaccessible otherwise.

3.3 | Piecewise polynomial control parameterization

While piecewise constant control discretizations are by far the most common parameterization for multiple shooting based MPC, formulation (2) can accommodate piecewise polynomial control parameterizations with degrees varying between shooting intervals. The k-th component of the control trajectory on a shooting interval $[\tau_0, \tau_e]$ could be parameterized by a polynomial

$$v_{n,k}(t) = \sum_{i=0}^{n_{\text{deg}}} u_{n,k,i} \cdot (t - \tau_0)^i \tag{6}$$

of degree n_{deg}, such that the discrete control input is given by $u_n = (u_{n,k,i})_{i=0,\dots,n_{\text{deg}},\, k=1,\dots,n_v}$. This parametrization offers more degrees of freedom and can in general better approximate the optimal continuous-time control trajectory. Higher order control parametrizations could thus allow the use of longer shooting intervals compared to a piecewise constant parametrization, while maintaining the approximation quality of the MPC feedback law.

The practical MPC consideration in (4) motivates using a constant control parameterization on the first interval $[0, \Delta t]$. In order to combine a constant control input on the first shooting interval with higher order control parameterizations on other shooting intervals, an MOCP formulation is thus necessary.

For linear parametrizations, simple bounds on the inputs can be satisfied everywhere by enforcing them at the start and end of each control interval. In contrast, for higher order polynomial parametrizations, even simple control bounds might be violated within the shooting intervals if only enforced at the boundary points. In order to avoid excessive use of such violations, one possibility is to enforce the control bounds at n_{pc} equidistant points within every shooting interval. For any fixed point $\bar{\tau} \in [t_n, t_{n+1}]$, we have

$$v_{n,k}(\bar{\tau}) = \sum_{i=0}^{n_{\text{deg}}} u_{n,k,i} \cdot (\bar{\tau} - t_n)^i. \tag{7}$$

which results in a linear inequality constraint with coefficients $(\bar{\tau} - t_n)^i$ for every intermediate point on which a control bound is enforced. Additionally, one could add smooth penalties on violations of the control bounds and integrate them over the shooting intervals, see Section 3.4.1.

3.4 | Cost-to-go approximation via closed-loop costing

The idea of *closed-loop costing* (CLC) is to apply a simple control law $\kappa : \mathbb{R}^{n_x} \to \mathbb{R}^{n_v}$ to the nonlinear system in (3c) and assign the resulting cost term as a terminal cost[50,51,52,11]. This control law is typically a locally stabilizing linear controller, e.g. corresponding to an LQR controller. In the Reinforcement Learning and Dynamic Programming community, this idea is more commonly referred to as rollout of a base policy[53]. The cost-to-go is approximated by the integrated cost associated with the

closed-loop system under policy κ. The closed-loop costing problem corresponding to (3) can be written as

$$\min_{x(\cdot),v(\cdot)} \quad \int_0^{T_1} \ell(x(t), v(t))\,\mathrm{d}t + \int_{T_1}^{T_{\mathrm{clc}}} \ell(x(t), \kappa(x(t)))\,\mathrm{d}t \tag{8a}$$

$$\text{s.t.} \quad 0 = x(0) - \bar{x}_0, \tag{8b}$$

$$0 = f(t, x(t), \dot{x}(t), v(t)), \qquad t \in [0, T_1), \tag{8c}$$

$$0 = f(t, x(t), \dot{x}(t), \kappa(x(t))), \; t \in [T_1, T_{\mathrm{clc}}), \tag{8d}$$

$$0 \geq g(x(t), v(t)), \qquad t \in [0, T_1], \tag{8e}$$

$$0 \geq g(x(t), \kappa(x(t))), \qquad t \in [T_1, T_{\mathrm{clc}}), \tag{8f}$$

$$0 = g_{\mathrm{e}}(x(T_{\mathrm{clc}})), \tag{8g}$$

where the first part of the horizon $[0, T_1]$ is the so-called *control horizon* and the latter part the *simulation horizon* or *closed-loop costing horizon*. Note that the control input is only defined on the control horizon, i.e. $v : [0, T_1] \to \mathbb{R}^{n_v}$ and the control dimension is zero on the CLC horizon. A discrete-time result showing that the stability region grows when increasing the CLC horizon is presented in the work by Magni and colleagues[54]. Note that the constraints are still imposed on the CLC horizon and violations correspond to infinite cost values[50].

Closed-loop costing problems can be expressed using two phases. In the terminal phase, the controls are replaced by the control law in the cost and dynamics expressions, resulting in a phase k with $n_{u,k} = 0$. In literature, single shooting has been the method of choice to handle the CLC horizon due to its superior computational efficiency in the case $n_u = 0$. However, the beneficial convergence properties also make multiple shooting an attractive option in `acados`: the partial condensing algorithm provided by `HPIPM` allows to condense blocks of an arbitrary number of stages, and this can be exploited to condense away all shooting nodes in the CLC horizon, while possibly retaining them in the control horizon. This makes the computational efficiency of multiple shooting similar to the one of single shooting, as the underlying QP solver does not see the shooting intervals corresponding to the CLC horizon. For the experiments in Section 5.1, it was necessary to use CLC with multiple shooting to achieve convergence. To the best of our knowledge, the approximate infinite horizon corresponding to the closed-loop costing phase was only implemented with a single shooting interval in previous works[50,51,52,11,55].

3.4.1 | Practical treatment of constraints with closed-loop costing

In practical OCP formulations state constraints are often replaced by penalties in the cost function to avoid infeasible OCPs[56]. Especially, when using longer shooting intervals, it is very important to accuaretely integrate the cost term, including penalties[9].

Accurate cost integration together with a propagation of the cost gradient and a Gauss-Newton Hessian can be performed efficiently. This is realized by the Gauss-Newton Runge-Kutta (GNRK) integrators and implemented in `acados`[9]. They can effectively handle L_2 constraint penalties. This implementation has been extended to treat more general convex-over-nonlinear cost terms of the form

$$\ell(x(t), v(t)) = \phi(r(x(t), v(t))) \tag{9}$$

with a smooth convex function ϕ and a nonlinear function r. The new implementation is able to integrate such cost terms together with their generalized Gauss-Newton (GGN) Hessian[57,58]. This is required to handle more general smooth penalties effectively, such as the squashed barrier closed-loop costing formulation described in Section 3.4.2.

Despite the reformulation of state constraints as penalties, even simple control bounds can render OCP (8) infeasible if the policy $\kappa(\cdot)$ would choose controls that violate those bounds on the CLC horizon. This would implicitly define a terminal region constraint. If one wants to avoid that, a blunt, but practical approach would be to not impose the control bounds on the CLC horizon, when using a linear control law κ. Another option is to replace control bounds on the CLC horizon with additional penalties. A third option is to combine the penalty approach with "squashing", as detailed next.

3.4.2 | Squashed closed-loop costing

The idea of squashed CLC is to create a CLC OCP which is aware of the control bounds on the CLC horizon. This can be achieved by defining the CLC control law as a squashed version of the linear control law, ensuring that control bounds are always satisfied. A function $\sigma : \mathbb{R} \to (-1, 1)$ is called *squashing function*, if it is twice continuously differentiable, strictly monotonically increasing, odd, and satisfies $\lim_{z\to\pm\infty} \sigma(z) = \pm 1$. Additionally, it is important, that $\frac{\partial\sigma}{\partial z}(0) = 1$, such that a linear control law is locally not changed. One example is $\sigma(z) = \tanh(z)$, which was also used in previous works[59].

In particular, a control variable u_i with symmetric bounds $[-\bar{u}_i, \bar{u}_i]$, can be reformulated using a new variable v and replacing u_i by the expression $\bar{u}_i \cdot \tanh(\frac{v}{\bar{u}_i})$ in the OCP.

It is recommended to use squashing together with barrier penalties such that an iterative solver does not go arbitrarily close to the squashed boundaries and gets stuck there[59]. A function $\beta : \mathbb{R} \to \mathbb{R}_{\geq 0} \cup \{\infty\}$ is called barrier function if it is twice continuously differentiable, strictly convex and $\lim_{z\to\pm 1} \beta(z) = \infty$. In particular, $\beta(z) = -\log(1 + z) - \log(-z + 1)$ is such a function and is used in this paper. The squashed LQR CLC variant with a progressively increasing barrier, can be interpreted in the framework of progressive tightening[60], which offers both, numerical benefits and closed loop stability guarantees for constrained nonlinear MPC. It is also possible to extend this concept to one-sided constraints by using a smooth-max function instead of the sigmoid as a squashing function and a one sided barrier.

3.5 | Partial tightening

The partial tightening concept was introduced by Zanelli et al.[61] as a strategy to reduce the computational cost of solving NMPC problems. To this end, an approximate formulation is introduced which divides the horizon into two phases. On the second phase, the tightened horizon, the constraints are tightened, i.e. replaced by barrier formulations. More precisely, a constraint $0 \geq g_j(x(t), v(t))$ as in (3) is replaced by a logarithmic barrier term

$$-\tau \log(g_j(x(t), v(t))), \tag{10}$$

which is added to the cost function with a barrier parameter τ.

The partial tightening formulation allows to derive closed-loop stability guarentees, based on the property that the stage cost are monotonically increasing with the stage index,[61]. This concept has been further extended to *progressive tightening*, which extends the concept to e.g. formulations with stage-wise increasing barrier parameters τ.

Partial and progressive tightening formulations have been analyzed in the literature regarding closed-loop stability properties[62,60] and successfully applied in practical applications such as collision avoidance for motion planning[63], control of a human-sized quadrotor[64], as well as voltage control in active distribution networks[65].

The barrier formulation in (10) can be combined with a series of monotonically increasing values for τ for subsequent stage costs on the tightened horizon to a progressive tightening formulation.

In context of the Real-Time Iteration (RTI), the variables corresponding to this second phase can be fully eliminated in the preparation phase using a Riccati recursion. In an SQP setting this can be achieved by performing partial condensing and summarizing all stages of the tightened horizon into a single block of the reduced QP.

4 | EFFICIENT IMPLEMENTATION IN `acados`

This section presents how MOCPs can be posed for an efficient solution in `acados`. Starting from problem (2), we discuss in Sections 4.1 and 4.2 the treatment of transition stages and how the problem fits into the internal formulation treated by `acados`. Section 4.3 presents a tutorial MOCP example and how it can be formulated for an efficient solution in `acados`.

4.1 | Treating transition stages

In order to incorporate transitions between two phases into an existing SQP software for OCP structured problems, we first consider the simple case where (2c) boils down to $x_{k+1,0} = x_{k,N_k}$, $n_{\eta,k} = 0$ and $E_k(\cdot) \equiv 0$. These trivial phase transitions can be eliminated by enforcing $\phi_{k,N_k-1}(x_{k,N_k-1}, u_{k,N_k-1}) = x_{k+1,0}$ and removing x_{k,N_k}, η_k from the problem.

In the nontrivial case, we observe that the transition equation (2c) has the same form as the equation for a shooting gap (2d). We thus call these *discrete transition stages*. On the discrete transition stage, the discrete decision variable η_k takes the role of a control input and E_k takes the role of the stage cost.

As opposed to formulating the transition as a concatenation of a dynamics step and a transition, the implementation of a transition with a separate discrete transition stage has the following advantages: Both the terminal state of the phase before the transition and the initial state of the phase after the transition are readily available in the solver and the transition cost E_k fits seamlessly into an MOCP formulation.

4.2 | Multi-phase multiple shooting formulation in acados

The discrete-time MOCP in (2) can be framed as a regular OCP-structured problem with stage-varying costs and constraints and where – in contrast to the standard setting – the control and state dimension might vary stage-wise,

$$\begin{aligned}
\underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{minimize}} \quad & \sum_{j=0}^{N} L_j(x_j, u_j, \Delta t_j) + E(x_N) && \text{(11a)}\\
\text{subject to} \quad & \bar{x}_0 = x(0), && \text{(11b)}\\
& x_{j+1} = \phi_j(x_j, u_j),\ j = 0, \dots, N-1, && \text{(11c)}\\
& 0 \geq h_j(x_j, u_j),\ j = 0, \dots, N-1. && \text{(11d)}
\end{aligned}$$

This OCP consists of N stages, which capture both the shooting intervals and the transition stages. In acados, nontrivial transitions are modeled by adding an extra phase comprised of a single interval. The transition function Γ_k is specified using the acados discrete dynamics module and the discrete decision variable η_k corresponds to the control variable for this phase. The number of stages N for the OCP (11) would thus be $N = M - 1 + \sum_{k=1}^{M} N_k$ if all transitions are nontrivial and $N = \sum_{k=1}^{M} N_k$ if all transitions are trivial. Note that the stage cost L_j does not depend on the shooting interval length if j is a transition stage.

A major challenge in tackling OCP (11) with efficient structure-exploiting solvers is that the dimensions of states, controls and constraints are varying arbitrarily between stages. In acados, this can be easily achieved by the different modules of the SQP type algorithm. Each stage has its own module to compute and linearize cost and constraint functions. Each gap constraint (11c) is associated with a module to evaluate and linearize it. The acados internal OCP-structured QP subproblem is based on the HPIPM software package[30]. In addition to interior-point solvers for dense and OCP-structured QP formulations, HPIPM offers efficient routines for transforming OCP-structured QPs into dense QPs via full condensing or smaller OCP-structured QPs via partial condensing[66]. The HPIPM core algorithms, full and partial condensing, and the Riccati-based QP solver, all support stage-varying dimensions. These full and partial condensing algorithms allow any dense QP solver, such as DAQP[32] and qpOASES[31] or HPIPM itself, as well as the OCP-structure exploiting solver in HPIPM, to be used seamlessly for multi-phase problem formulations in acados.

4.3 | Tutorial example of a multi-phase OCP

In this section, we discuss a tutorial example of an MOCP which uses models on distinct parts of the horizon and detail how to formulate the problem in acados. The code to reproduce the figures in this section is publicly available[†]. We regard a double integrator system which consists of the state $\xi_1 = (p, s)$, with position p and speed s. The control input v_1 is the acceleration a and the continuous-time dynamics are simply given by

$$0 = f_1(\xi_1, \dot{\xi}_1, v_1) = \begin{bmatrix} \dot{p} - s \\ \dot{s} - a \end{bmatrix}.$$

Let us formulate an MOCP with two phases where an approximate model is employed in the second phase. The approximate model does not consider acceleration and regards the velocity as the control input instead yielding a one dimensional system, i.e.,

[†] https://github.com/acados/acados/blob/master/examples/acados_python/mocp_transition_example/main.py

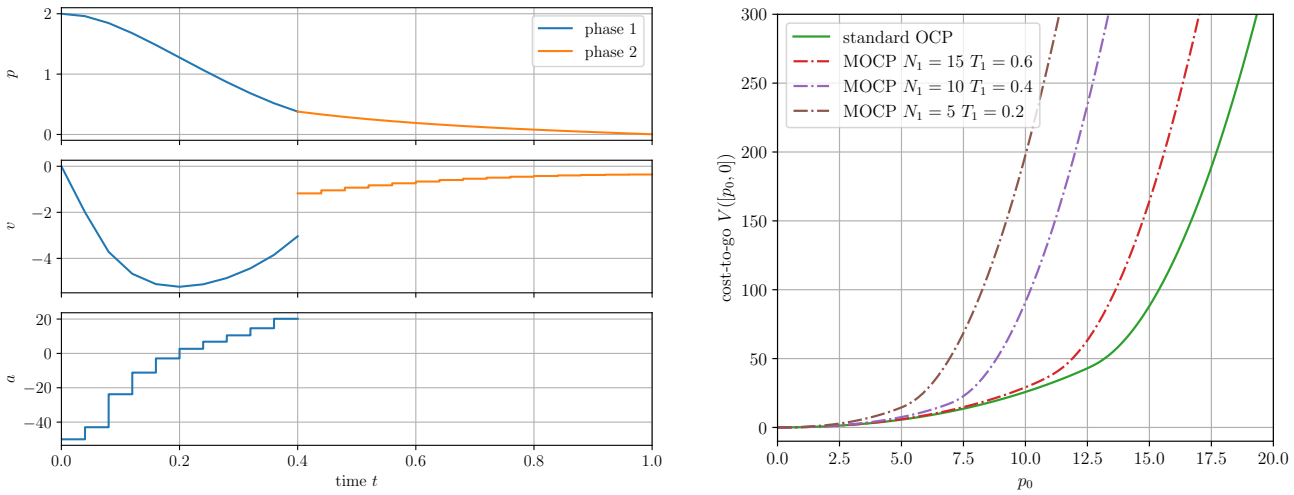

FIGURE 1 Tutorial example: We consider a double integrator, which is in the second phase approximated by a single integrator model. The left plot shows the state and control trajectories for the tutorial MOCP. The right plot shows the cost-to-go function as a function of the initial position p_0 for different approximations of the OCP which only uses the double integrator model.

$\xi_2 = p$, $v_2 = s$. The system dynamics are then $f_2(\xi_2, \dot{\xi}_2, v_2) = \dot{p} - s$. The transition function is $\Gamma_k(\xi_1) = p$. We define an MOCP with a time horizon of $T = 1$ and the cost functions

$$\ell_1(\xi_1, v_1) = p^2 + 0.1s^2 + 10^{-3}a^2, \qquad E_1(\xi_1) = p^2 + 0.01s^2,$$
$$\ell_2(\xi_1, v_2) = p^2 + 0.1s^2, \qquad E_2(\xi_2) = 10p^2.$$

We impose control bounds as constraints on both phases: In the first phase, we have $-50 \leq a \leq 50$, in the second phase, we use $-5 \leq s \leq 5$. We associate the first phase with the interval $[0, 0.4]$ and the second phase with $[0.4, 1.0]$ and divide the both intervals uniformly into 10 and 15 shooting intervals, respectively. The resulting problem is a QP and solved with a single SQP iteration in `acados`. The optimal trajectory for $\bar{x}_0 = [2, 0]^\top$ is visualized in the left plot in Figure 1.

The right plot in Figure 1 shows how MOCPs can approximate the cost-to-go function. All OCPs use $N = 25$ shooting intervals and a time horizon of $T = 1$. The MOCPs use the single integrator approximate OCP phase as described above on the interval $[T_1, T]$. The shooting intervals are equidistant on both phases with N_1 intervals in the first phase and $N - N_1$ in the second phase.

We want to use the tutorial example to demonstrate the formulation of an MOCP with a nontrivial transition using the new `acados` interface. Firstly, we need to define the models for all phases, the double and the single integrator model as well as the transition model. The single integrator model is defined as an explicit ODE using `CasADi` as follows

```
import casadi as ca
def get_single_integrator_model() -> AcadosModel:
  model = AcadosModel()
  model.name = 'single_integrator'
  model.x = ca.SX.sym('p')
  model.u = ca.SX.sym('v')
  model.f_expl_expr = model.u
  return model
```

The transition model is defined as

```
def get_transition_model() -> AcadosModel:
  model = AcadosModel()
  model.name = 'transition_model'
  p = ca.SX.sym('p')
  v = ca.SX.sym('v')
  model.x = ca.vertcat(p, v)
  model.disc_dyn_expr = p
  return model
```

Let's assume that we already formulated the OCPs for the individual phases, the one with the single and double integrator model, which use the established Python interface of `acados`, more precisely the `AcadosOcp` class.

```
def formulate_double_integrator_ocp() -> AcadosOcp:
  ...
def formulate_single_integrator_ocp() -> AcadosOcp:
  ...
```

A multi-phase OCP can be formulated in acados using the AcadosMultiPhaseOcp class, which is created by specifying the number of stages per phase.

```
N_list = [10, 1, 15]  # 10 stages for double integrator, 1 stage for transition, 15 stages for single integrator
ocp = AcadosMultiphaseOcp(N_list=N_list)
```

The novel interface allows to define the dynamics, cost and constraints of one phase utilizing the single-phase OCP formulations, i.e. the AcadosOcp class.

```
# define phases for single and double integrator
phase_0 = formulate_double_integrator_ocp()
ocp.set_phase(phase_0, 0)
phase_2 = formulate_single_integrator_ocp()
ocp.set_phase(phase_2, 2)
# define the transition phase and cost
phase_1 = AcadosOcp()
phase_1.model = get_transition_model()
phase_1.cost.cost_type = 'NONLINEAR_LS'
phase_1.model.cost_y_expr = phase_1.model.x
phase_1.cost.W = np.diag([L2_COST_P, 1e-1 * L2_COST_V])
phase_1.cost.yref = np.array([0., 0.])
ocp.set_phase(phase_1, 1)
```

Most solver options can be set in the same way as for the single-phase OCP:

```
ocp.solver_options.nlp_solver_type = 'SQP'
ocp.solver_options.time_steps = np.array(N_list[0] * [0.4/N_list[0]]
+ [1.0] # transition stage
+ N_list[2] * [0.6 / N_list[2]])
```

Some additional solver options that can not vary for single-phase OCP problems can be set additionally.

```
ocp.mocp_opts.integrator_type = ['ERK', 'DISCRETE', 'ERK']
```

Finally, an AcadosOcpSolver can be created from the AcadosMultiphaseOcp, just as from a AcadosOcp object.

```
acados_ocp_solver = AcadosOcpSolver(ocp)
```

The interactions with the solver are independent of whether it was created from a single or multi-phase OCP formulation.

5 | NUMERICAL EXPERIMENTS

This section presents three numerical case studies. First, Section 5.1 compares MPC controllers with different control parameterizations and closed-loop costing variants on an inverted pendulum test problem. Second, Section 5.2 regards the task of controlling a differential drive robot directly through the voltages of actuators and compares the performance of controllers based on single- and multi-phase problem formulations. Third, Section 5.3 shows the efficiency of partial tightening with real-time iterations qualitatively replicating the benchmark results from the first partial tightening paper[61]. All experiments have been carried out using acados v0.3.2 via its Python interface on a Laptop with an Intel i5-8365U CPU, 16 GB of RAM running Ubuntu 22.04.

Note that these experiments only compare acados controllers based on single- and multi-phase OCP formulations. Comparisons with solvers based on other software packages are out of the scope of this paper. However, since the computation times of single- and multi-phase OCP formulations within acados are consistent, results from existing software comparisons can be transferred to multi-phase problems if all competing solvers support MOCPs.

5.1 | Inverted pendulum on cart test problem

We investigate the inverted pendulum on cart problem in the setting from the benchmark presented by Frey et al.[9] and compare different controllers with different closed-loop costing variants and control parametrizations. The code of the orignal benchmark has been adapted to incorporate the new controllers[‡].

[‡] https://github.com/FreyJo/GNRK_benchmark

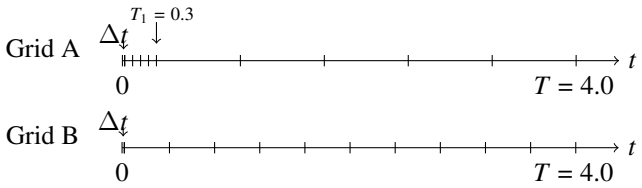

FIGURE 2 Two time grids considered in the benchmark. Both grids start with an interval of length $\Delta t = 0.02$. The grids are visualized for $N = 10$ shooting intervals. For Grid A, the interval $[\Delta t, T_1]$ is divided into $N_1 = 4$ equidistant intervals, and $[T_1, T]$ into $N_2 = N - N_1 - 1 = 5$ equidistant intervals. For Grid B, the interval $[\Delta t, T]$ is divided into $N - 1$ intervals.

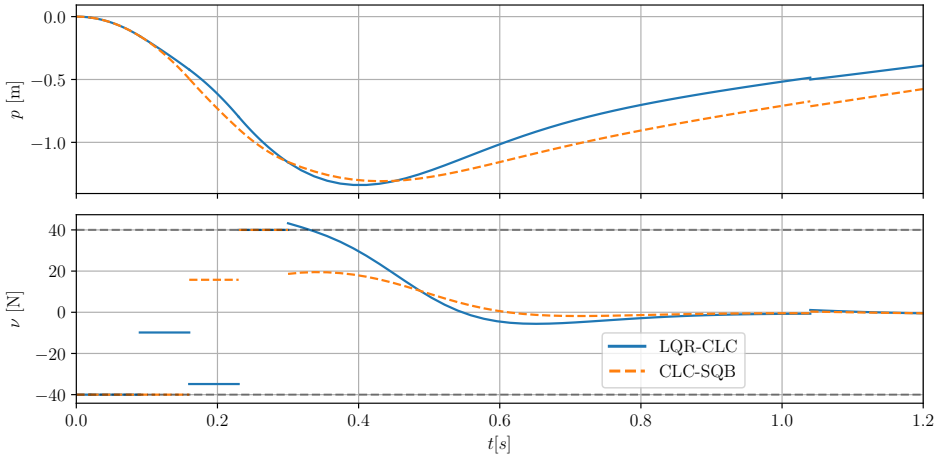

FIGURE 3 Multiple-shooting with closed-loop costing. The solver iterates after performing 3 iterations on the first problem in the closed-loop scenario in Figure 4 are visualized.

The differential state of the model is $x = [p, \theta, s, \omega]^\top$ with cart position p, cart velocity s, angle of the pendulum θ and angular velocity ω. The control input v is a force acting on the cart in the horizontal plane. The system dynamics can be found e.g. in [12]. In our OCP formulation, v is constrained to be in $[-40, 40]$. The simulation starts with an initial state $\bar{x}_0 = [0, \frac{\pi}{5}, 0, 0]^\top$. The goal is to drive all states to zero, i.e. the unstable upright position. We formulate the following nonlinear least-squares cost consisting of quadratic costs on states and controls and a term penalizing a position p outside of $[-1, 1] =: [p_{\min}, p_{\max}]$, namely

$$l(x, u) = x^\top Q x + v^\top R v + \gamma \cdot (\max(p_{\min} - p, 0))^2 + \gamma \cdot (\max(p - p_{\max}, 0)^2$$

where the cost weights are chosen as $\gamma = 5 \cdot 10^4$, $Q = \mathrm{diag}(100, 10^3, 0.01, 0.01)$, $R = 0.2$. The terminal cost term is set to $M_{\text{pend}}(x) = x^\top P x$, where P is obtained as solution of the discrete algebraic Riccati equation with cost and dynamics linearized at the steady state.

The controllers use two different discretization grids, visualized in Figure 2. The plant is simulated with a time step of $\Delta t = 0.02$s. All controllers use the GNRK cost discretization described in our recent paper [9] and Section 3.4.1 with a Gauss-Radau IIA method of order 7 on each shooting interval. Additionally, we compare various control parameterizations, namely piecewise polynomials of different degree on all but the first shooting interval $[0, \Delta t]$, see Section 3.3, and closed-loop costing formulation, see Section 3.4. In order to conveniently formulate the models with piecewise polynomial controls, we added the functionality `AcadosModel.reformulate_with_polynomial_control()`.

An overview of all controller variants considered is presented in Table 1, some of the closed-loop trajectories are plotted in Figure 4 and some of the open-loop control trajectories corresponding to the first problem are shown in Figure 5.

TABLE 1 Closed-loop comparison of different controller variants on the pendulum test problem with different discretization grids and the control parametrizations, including closed-loop-costing and piecewise polynomial controls.

variant ID	N	Grid	control parametrization	comp. time / iter [ms]	relative suboptimality [%]
IDEAL	200	B	pw. constant	3.69	0.00
REF	20	B	pw. constant	0.43	3.72
REF-N10	10	B	pw. constant	0.21	267.57
PW-LIN-B	10	B	pw. polynomials with $n_{\text{deg}} = 1, n_{\text{pc}} = 2$	0.23	13.54
PW-CUBIC-B	10	B	pw. polynomials with $n_{\text{deg}} = 3, n_{\text{pc}} = 10$	0.24	13.59
LQR-CLC	10	A	unconstrained LQR CLC	0.22	3.41
CLC-SQB	10	A	Squashed + prog. barrier CLC	0.23	5.13
PW-CONST-A	10	A	pw. constant	0.22	0.49
PW-LIN-A	10	A	pw. polynomials with $n_{\text{deg}} = 1, n_{\text{pc}} = 2$	0.23	0.40
PW-QUAD-1	10	A	pw. polynomials with $n_{\text{deg}} = 2, n_{\text{pc}} = 4$	0.24	0.06
PW-QUAD-2	10	A	pw. polynomials with $n_{\text{deg}} = 2, n_{\text{pc}} = 10$	0.26	0.15
PW-CUBIC-1	10	A	pw. polynomials with $n_{\text{deg}} = 3, n_{\text{pc}} = 4$	0.23	0.07
PW-CUBIC-2	10	A	pw. polynomials with $n_{\text{deg}} = 3, n_{\text{pc}} = 6$	0.24	0.03
PW-CUBIC-3	10	A	pw. polynomials with $n_{\text{deg}} = 3, n_{\text{pc}} = 10$	0.26	0.02

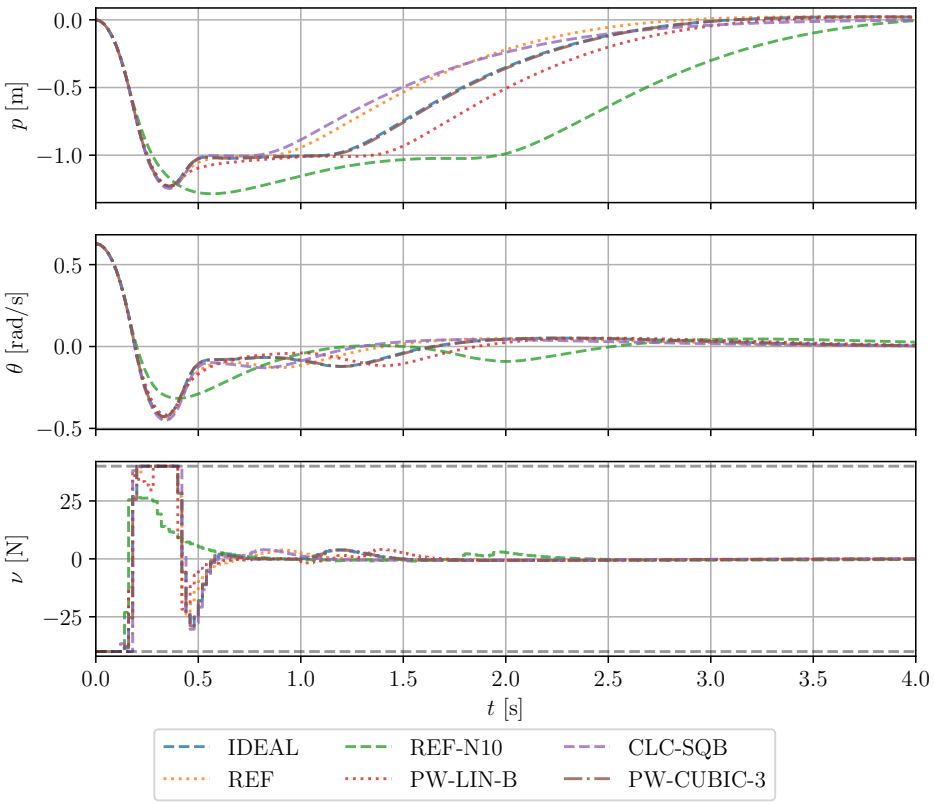

FIGURE 4 Closed-loop trajectories corresponding to the benchmark results in Table 1. Values for the postion p outside of $[-1, 1]$ are heavily penalized and result in high suboptimality, which is reported in Figure 6 and Table 1.

5.1.1 | Controller variants

The controller labeled IDEAL uses a uniform time discretization corresponding to time steps Δt with piecewise constant controls, i.e. no mismatch between the plant and the OCP model. This controller gives the best closed-loop performance, but has the highest computational complexity. The controller REF corresponds to the variant which gave the best tradeoff between computation time and closed-loop performance in the original benchmark[9]. The goal is to derive a discrete-time OCP variant

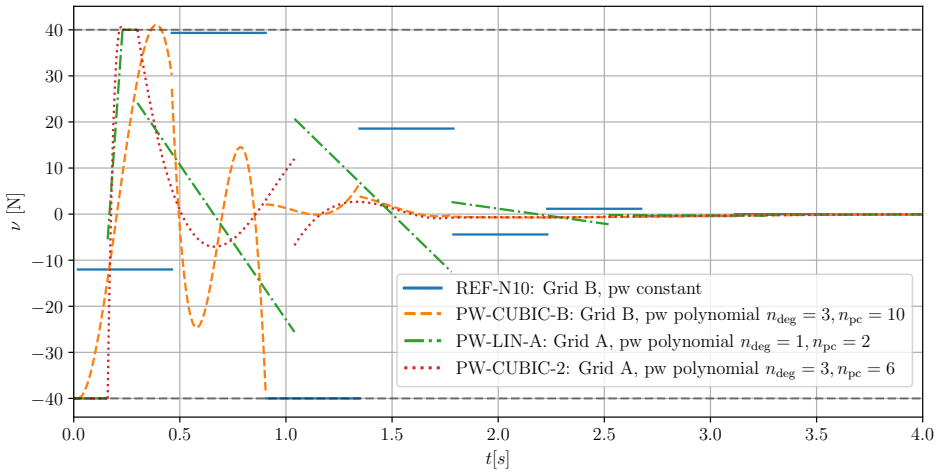

FIGURE 5 Open-loop control trajectories corresponding to the first problem solved in the closed-loop simulation visualized in Figure 4. Details on the controller variants are in Table 1.

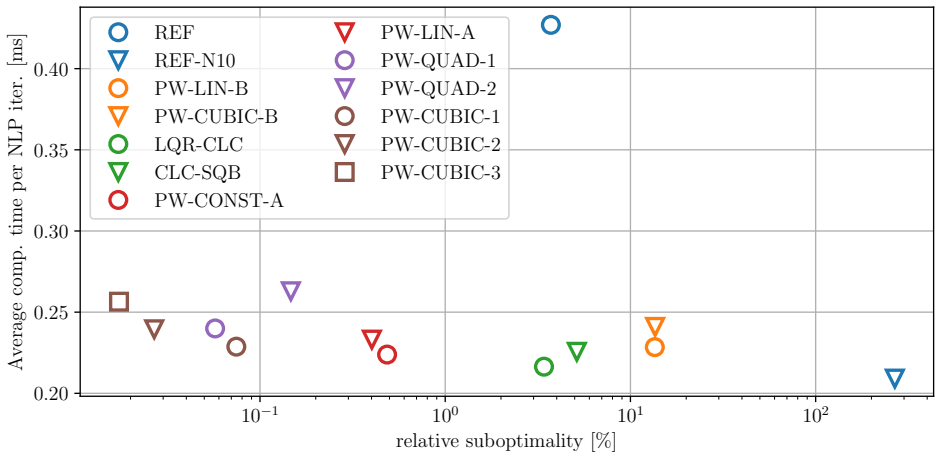

FIGURE 6 Pareto plot comparing the controllers in Table 1 in terms of average computation time per NLP solver iteration and relative suboptimality with respect to the controller IDEAL which is not included in this Figure.

which uses less shooting intervals than this controller in order to reduce the associated computational complexity, while attaining similar closed-loop performance.

The variant REF-N10 uses the same kind of grid, but divides the interval $[\Delta t, T]$ into 9 instead of 19 shooting intervals. Thus, the computational complexity associated with one iteration is roughly halved. However, the closed-loop performance is highly degraded. The controller variant PW-LIN-B has only one difference, which is to use a piecewise linear control parametrization on all but the first shooting interval. This results in a much better performance compared to REF-N10, with marginally higher computational burden, indicating that the cost-to-go approximation on this coarse grid is limited by the control parameterization.

We evaluate two closed-loop costing (CLC) variants. Both CLC controllers use Grid A, where the second part of the time horizon, $[T_1, T]$ is implemented with a second phase, see Section 3.4. For the LQR-CLC controller, the control law is the stabilizing LQR control law $\kappa_{\text{LQR}}(x) = Kx$. The control input constraints are not enforced on the CLC horizon. This controller results in a closed-loop performance that is similar to the one of controller REF proposed in the original benchmark[9] at half the compuation time.

The other CLC controller is labeled CLC-SQB and uses the same linear feedback law with additional squashing to impose the given control limits, $\kappa_{\text{squash}}(x) = \sigma(Kx)$. Additionally, a barrier term $\beta(\sigma(Kx))$ is added on the CLC horizon. The resulting controller can therefore not plan with violations of the control bounds in the CLC horizon. This controller yields slightly worse

closed-loop performance compared to the standard LQR-CLC controller. We attribute this observation to the fact that the CLC controller with squashing naturally has a more conservative plan, which results in some additional suboptimality. Figure 3 visualizes intermediate iterates of the controllers LQR-CLC and CLC-SQB and gives some insights into CLC. It can be seen that the controls on the CLC horizon are continuous, since κ and the state trajectory are continuous. However, for the intermediate iterates of the multiple-shooting formulation, the shooting gaps, which are visible in Fig. 3 result in gaps in the controls within the CLC horizon which are closed at convergence. Lastly, we observe that the LQR-CLC controller violates the control bounds in its plan, while the squashed variant CLC-SQB inherently respects them.

As an alternative to the CLC controllers, some variants are included, which use discretization Grid A, but a standard control horizon on the second part of the grid $[T_1, T]$ and piecewise polynomial controls on $[\Delta t, T]$ with the control bounds enforced on n_{pc} equidistant points on every shooting interval, see Section 3.3.

The controllers PW-CONST-A and PW-LIN-A exactly enforce the control bounds, are among the controllers with the lowest computational complexities and result in a relative suboptimality below 0.5 %. In particular, they are able to roughly halve the computational complexity of REF, the best controller in the original benchmark[9], while reducing the suboptimality by a factor greater than 7.

5.1.2 | Discussion

The fact that controller PW-CONST-A has a lower computational burden compared to the CLC variant is attributed to the fact that the control law $\kappa(\cdot)$ and its derivatives have to be evaluated often within every step and Newton iteration of the implicit integrator. The additional computational burden of a larger input dimension, which is 0 or 1 on the latter part of the horizon, is rather small for efficient state-of-the-art QP solvers.

There are certainly other examples in which a CLC controller is computationally more attractive. In addition to the test setting, this of course depends on the underlying software framework for linearization and solution of the subproblems. In particular, for modern OCP-structure exploiting software frameworks, the computational burden associated with an extra control variable is less significant compared to when CLC was introduced in the early 1990s[28].

A similar comparison of CLC controllers with respect to obvious alternatives is not to be found in the existing literature on CLC[50,51,52,11,55]. In addition to the LQR-CLC controller described above, we compare a related variant, which uses $N_2 = 1$, i.e. single shooting on the CLC horizon, and instead performs five steps of IRK on this interval. This controller variant does not converge in the first simulation step of the scenario and has a computational complexity similar to LQR-CLC. This indicates that the multiple shooting CLC implementation proposed in this paper has desirable properties compared to the single-shooting CLC variants used in previous works.

Overall, the comparison of CLC with respect to nonuniform OCP discretizations shows that the latter are very competitive. On this test example, the nonuniform grid variant outperforms the LQR closed-loop costing based controller in terms of closed-loop performance by a factor of 7 while requiring an equal amount of computational resources.

Finally, we regard controllers with piecewise polynomial control parameterizations of degree $n_{\mathrm{deg}} > 1$ which enforce the control bounds on n_{pc} intermediate points. The open-loop solutions corresponding to the first problems within the closed-loop simulation are visualized for a few controllers in Figure 5. We observe small violations of the open-loop control trajectories with respect to the control bounds, e.g. at $t \approx 0.2$ for PW-CUBIC-2 and $t \approx 0.35$ for PW-CUBIC-B.

All variants in Figure 5 use 10 shooting intervals. Comparing REF-N10 in Figure 5 with the optimal closed-loop control trajectory in Figure 4, we observe that this planned controls are qualitatively very different from the optimal ones, resulting in a bad cost-to-go approximation and thus closed-loop performance. In contrast, PW-CUBIC-2 can capture the qualitative behavior of the optimal closed-loop control trajectory well. In particular, the relatively short intervals in the first part of the horizon enable the cubic polynomials to approximate the optimal bang-bang solution in this phase well. In contrast, the variant PW-CUBIC-B, which uses the same control parametrization as PW-CUBIC-2, but on grid A, approximates this optimal bang-bang behavior with a slower transition, see Figure 4, resulting in significantly higher closed-loop cost, see Table 1. A visual comparison of the open-loop trajectories of PW-CUBIC-2 and PW-LIN-A shows that the optimal bang-bang behavior in the first part of the horizon is approximated similarly, while on the first long interval, starting at $t = 0.3$, the cubic polynomials capture the optimal behavior qualitatively better, as piecewise linear functions do not have an inflection point.

When comparing controllers with the same parametrization, but different grids on which the control bounds are enforced, i.e. varying n_{pc}, we can observe two effects. Firstly, comparing PW-QUAD-1 and PW-QUAD-2, we see that enforcing control bounds on a finer grid can result in worse closed loop performance. This can be attributed to the fact that enforcing the control

bounds everywhere using polynomials results in a more conservative approximation of the cost-to-go, since the planned controls are approximated within the bounds. On the other hand, regarding PW-CUBIC-1, PW-CUBIC-2, PW-CUBIC-3, we observe that enforcing constraints on a tighter grid can improve the approximation quality.

Overall, the general trend in our experiments shows that additional degrees of freedom in piecewise polynomial control parametrizations can result in a strongly improved closed-loop performance, if the control bounds are enforced on a sufficiently fine grid, which we attribute to a better approximation of the closed-loop cost. In particular, PW-CUBIC-3 results in better performance than PW-QUAD-2, which in turn improves on PW-LIN-A.

Of course one needs to be careful, when it comes to drawing general conclusions from the presented results on piecewise polynomial control parameterizations. The additional computational cost of polynomial control parameterizations of higher degrees and handling the corresponding bounds, will be much more significant, with growing n_v. In addition to the dimensions, the optimal choice of control parameterization and where to enforce control bounds depends on the qualitative behavior of the continuous-time optimal solution (bang-bang or continuous), the time discretization grid, the available solvers and computational resources. However, the possibility of formulating such problems within an efficient software package can significantly improve the closed-loop performance of MPC controllers.

5.2 | Differential drive robot with actuator model and economic cost

We consider a differential drive robot and develop an NMPC controller which takes the underlying actuators into account allowing it to consider their power consumption in the cost function. The code to reproduce the results presented in this section is publicly available[§]

5.2.1 | Modelling

This subsection describes the differential drive model with and without actuators[67]. For simplicity, the dynamic model which disregards the actuators is presented first. This model consists of the state vector $x_{\text{simple}} = [p_{\text{x}}, p_{\text{y}}, v, \theta, \omega]$, where $p_{\text{x}}, p_{\text{y}}$ denotes the robots position in x- and y-coordinates, v the velocity of the robot, θ the heading angle, and ω the angular velocity. The input of this model are $u_{\text{simple}} = [\tau_{\text{r}}, \tau_{\text{l}}]$, the torques applied to the right and left driving wheel, respectively, in Nm. The evolution of the system is described by the ODE

$$\dot{x}_{\text{simple}} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \frac{a_1 + m_{\text{c}} d\omega^2}{m + a_2} \\ \frac{La_3 - m_{\text{c}} d\omega v}{I + L^2 a_2} \end{bmatrix}, \tag{12}$$

where the shorthands $a_1 = \frac{\tau_{\text{r}}+\tau_{\text{l}}}{2}$, $a_2 = \frac{2I_{\text{w}}}{R^2}$, $a_3 = \frac{\tau_r - \tau_l}{R}$ are introduced. The model contains the following parameters, which we assume to be constant: The total mass of the robot $m = 220\text{kg}$, the mass of the robot without the wheels and the rotating parts of the actuators $m_{\text{c}} = 200\text{kg}$, the robot's moment of inertia around the center of mass $I = 9.6\,\text{kg}\cdot\text{m}^2$, the combined moment of inertia about the wheel's axis of a driving wheel and the rotating part of the actuator $I_{\text{w}} = 0.1\text{kg}\cdot\text{m}^2$.

In addition, we consider a more accurate model of the robot which takes the two actuators, namely a motor on each driving wheel into account. The actuator model consists of the state $x_{\text{act}} = [p_{\text{x}}, p_{\text{y}}, v, \theta, \omega, I_{\text{r}}, I_{\text{l}}]$, where $I_{\text{r}}, I_{\text{l}}$ are the currents in the motor driving the right and left wheel, respectively, and the control input $u_{\text{act}} = [V_{\text{r}}, V_{\text{l}}]$, where $V_{\text{r}}, V_{\text{l}}$ denote the voltages applied to the motors. The evolution of the states common for both models is given by (12), where the motor torques $\tau_{\text{r}}, \tau_{\text{l}}$ are substituted by $\tau_{\text{r}} = K_1 I_{\text{r}}$ and $\tau_{\text{l}} = K_2 I_{\text{l}}$. In addition, the dynamics of the accurate model are given by

$$\begin{bmatrix} \dot{I}_{\text{r}} \\ \dot{I}_{\text{l}} \end{bmatrix} = \begin{bmatrix} -\frac{K_1\psi_1 - R_{\text{act}} I_{\text{r}} + V_{\text{r}}}{L_{\text{act}}} \\ -\frac{K_2\psi_2 - R_{\text{act}} I_{\text{l}} + V_{\text{l}}}{L_{\text{act}}} \end{bmatrix} \tag{13}$$

[§] https://github.com/FreyJo/ocp_solver_benchmark/blob/main/experiments/actuator_diff_drive.py

with

$$\psi_1 = \frac{\dot{p}_\mathrm{x}\cos\theta + \dot{p}_\mathrm{y}\sin\theta + L\omega}{R}, \tag{14}$$

$$\psi_2 = \frac{\dot{p}_\mathrm{x}\cos\theta + \dot{p}_\mathrm{y}\sin\theta - L\omega}{R}. \tag{15}$$

This accurate model additionally contains the motor constants $K_1 = 1.0$, $K_2 = 1.0$, the coil inductance $L_\mathrm{act} = 10^{-4}\mathrm{H}$, the coil resistance $R_\mathrm{act} = 0.05\Omega$ as parameters which we assume to be constant.

5.2.2 | Optimal control problem formulation

We want to minimize the following cost term

$$l_\mathrm{act}(x_\mathrm{act}, u_\mathrm{act}) = {x_\mathrm{act}}^\top Q x_\mathrm{act} + |V_\mathrm{r} I_\mathrm{r}| + |V_\mathrm{l} I_\mathrm{l}| \tag{16}$$

with $Q = \mathrm{diag}(10^3, 10^3, 10^{-4}, 1, 10^{-3}, 0.5, 0.5)$. Note that the second summand $|V_\mathrm{r} I_\mathrm{r}| + |V_\mathrm{l} I_\mathrm{l}|$ is an economic cost term corresponding to the power consumption. It is implemented in acados by introducing slack variables s_low, s_up and constraints $s_\mathrm{low} \leq [V_\mathrm{r} I_\mathrm{r}, V_\mathrm{l} I_\mathrm{l}]^\top \leq s_\mathrm{up}$ and replacing the term $|V_\mathrm{r} I_\mathrm{r}| + |V_\mathrm{l} I_\mathrm{l}|$ in the cost with $s_{\mathrm{low},1} + s_{\mathrm{low},2} + s_{\mathrm{up},1} + s_{\mathrm{up},2}$. When using the actuator model, for the full horizon, we use the terminal cost term:

$$E_\mathrm{act}(x_\mathrm{act}) = {x_\mathrm{act}}^\top Q x_\mathrm{act} \tag{17}$$

We want to investigate approximate MOCP formulations which use the simple differential drive model on the latter part of the horizon. The cost term in (16) can be approximated using the simple model by disregarding the economic cost term, i.e. using

$$l_\mathrm{simple}(x_\mathrm{simple}, u_\mathrm{simple}) = {x_\mathrm{simple}}^\top \tilde{Q} x_\mathrm{simple} + {u_\mathrm{simple}}^\top \tilde{R} u_\mathrm{simple} \tag{18}$$

with $\tilde{Q} = \mathrm{diag}(10^3, 10^3, 10^{-4}, 1, 10^{-3})$, $\tilde{R} = \mathrm{diag}(0.5, 0.5)$. As a terminal cost of the first phase with the actuator model, cf. (2), we define

$$E_\mathrm{trans}(x_\mathrm{act}) = 0.5\Delta t_\mathrm{trans} \cdot ({I_\mathrm{r}}^2 + {I_\mathrm{l}}^2), \tag{19}$$

where Δt_trans denotes the length of the last shooting interval on which the actuator model is used. This cost term corresponds to the tracking part of the cost in (16) and is needed since I_r, I_l are not included in the simple model. When using the simple model at the end of the horizon, we use the terminal cost term:

$$E_\mathrm{simple}(x_\mathrm{simple}) = {x_\mathrm{simple}}^\top \tilde{Q} x_\mathrm{simple}. \tag{20}$$

For both models, we impose the following path constraints on the state

$$0 \leq v \leq 1, \tag{21}$$

$$-0.5 \leq \omega \leq 0.5. \tag{22}$$

Additionally, we impose the following constraints on the control inputs when using the actuator model

$$-10 \leq V_\mathrm{l} \leq 10, \tag{23}$$

$$-10 \leq V_\mathrm{r} \leq 10. \tag{24}$$

Respectively, for the simple model, we impose

$$-60 \leq \tau_\mathrm{r} \leq 60, \tag{25}$$

$$-60 \leq \tau_\mathrm{l} \leq 60. \tag{26}$$

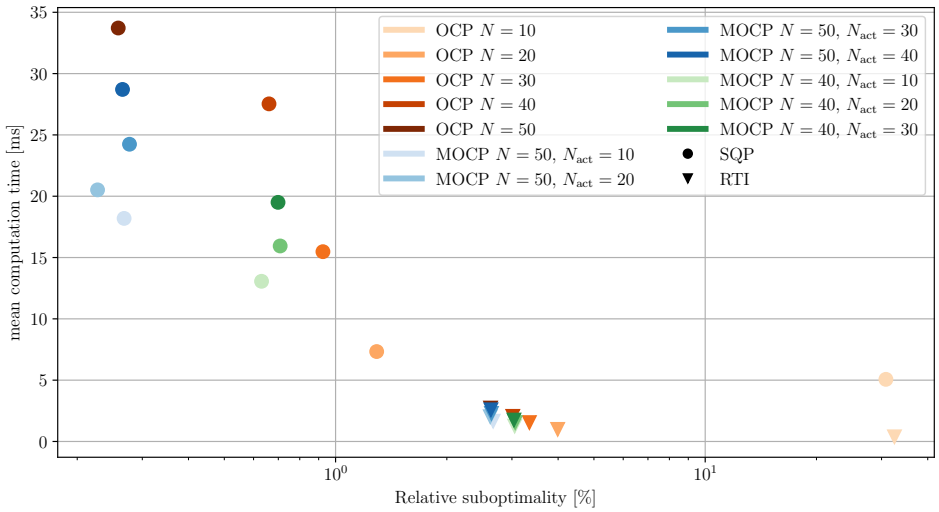

FIGURE 7 Pareto plot comparing different controller variants for directly controlling the actuators of a differential drive robot with an economic cost. All MOCP variants use $N = 50$ shooting intervals and $N - N_{act}$ of an approximate model which disregards the actuator dynamics.

5.2.3 | Comparing controller variants

We compare different controller variants. All variants use discretization steps of $\Delta t = 0.1$s, the model dynamics are integrated using an implicit Runge-Kutta method with a Legendre Butcher tableau of order 6 on each shooting interval. The controllers are based on an `acados` full-step SQP solver using a Gauss-Newton Hessian approximation and solving the QPs with `HPIPM` without condensing.

We consider MOCP variants with $N = 40$ and $N = 50$ shooting intervals and vary N_{act}, the number of shooting intervals on which the actuator model is used, afterwards, we use a transition stage and $N - N_{act}$ shooting intervals with the simple model. For the single-phase OCP variants, we vary the number of shooting intervals N, as this formulation does not allow for a transition.

We compare several controllers in a closed-loop simulation of 20s using the actuator model starting at the initial state $x_0 = (1, 1, 0, \pi, 0, 0, 0)$, exactly integrating the cost in (16) using 10 integrator steps for a sampling time of 0.1s. In order to evaluate relative suboptimality, a reference controller with $N = 60$ shooting intervals of the actuator model is used. Controllers with fully converged SQP and RTI variants, which only solve a single QP subproblem are evaluated in the same plot.

The Pareto plot in Figure 7 visualizes the suboptimality and computation time of the various variants. We observe that the Pareto front is dominated by the MOCP variants. Note that the variant OCP $N = 50$ correspond to the controller MOCP $N = 50, N_{act} = 50$. The solver variants OCP $N = 30$ and MOCP $N = 50, N_{act} = 10$ with SQP require a similar computation time, however the MOCP variant results in a three fold lower relative suboptimality. On the other hand, MOCP with $N = 50, N_{act} = 10$ delivers a similar suboptimality compared to the variant OCP $N = 50$, while only requiring roughly half of the computation time. Overall, the MOCP variants dominate the Pareto front in large parts and it is of course possible to generate many more combinations using MOCP formulations or by varying the number of SQP iterations. These results show that deriving an approximate OCP formulation and using it within an MOCP can result in an NMPC controller which is able to outperform NMPC controllers that only use a single-phase OCP formulation.

5.3 | Partial tightening

This section briefly demonstrates how partial tightening can be efficiently implemented within `acados`, qualitatively reproducing the results from the benchmark in the paper that introduced partial tightening[61]. To this end, we regard the inverted pendulum on cart model as in Section 5.1. The cost function is of linear least-squares form

$$l(x, v) = x^\top Q x + v^\top R v, \tag{27}$$

TABLE 2 Performance overview of different controllers with RTI and partial tightening. Maximum timings are over the closed-loop simulation in Figure 9 and are given in [ms] and relative suboptimality is evaluated by comparing with the controller variant $N = 100, N_{\text{exact}} = 100$.

	computation times			
Variant	preparation	feedback	total	rel. subopt. %
$N = 100, N_{\text{exact}} = 5$	0.23	0.32	0.54	21.25
$N = 100, N_{\text{exact}} = 10$	0.25	0.36	0.59	16.61
$N = 100, N_{\text{exact}} = 20$	0.24	0.44	0.65	12.25
$N = 100, N_{\text{exact}} = 50$	0.25	1.15	1.36	5.15
$N = 100, N_{\text{exact}} = 100$	0.23	2.87	3.05	0.00
$N = 50, N_{\text{exact}} = 50$	0.10	2.17	2.27	990.43

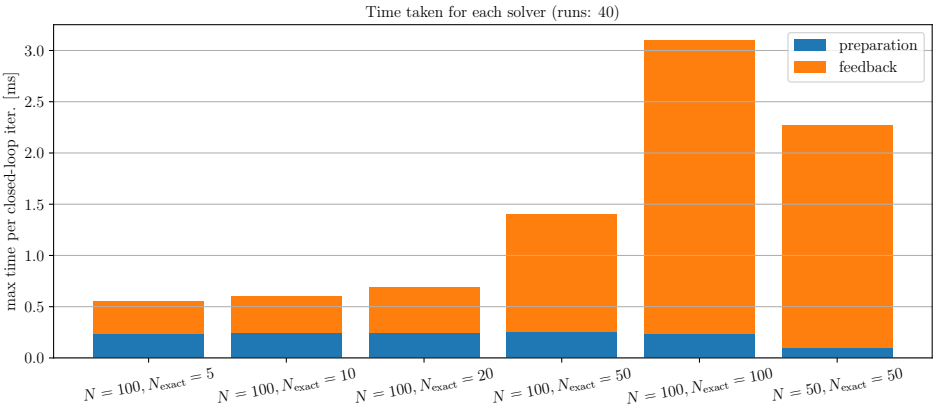

FIGURE 8 Maximum timings for the preparation and feedback phase over the closed-loop simulation shown in Figure 9. The timings are also given in Table 2.

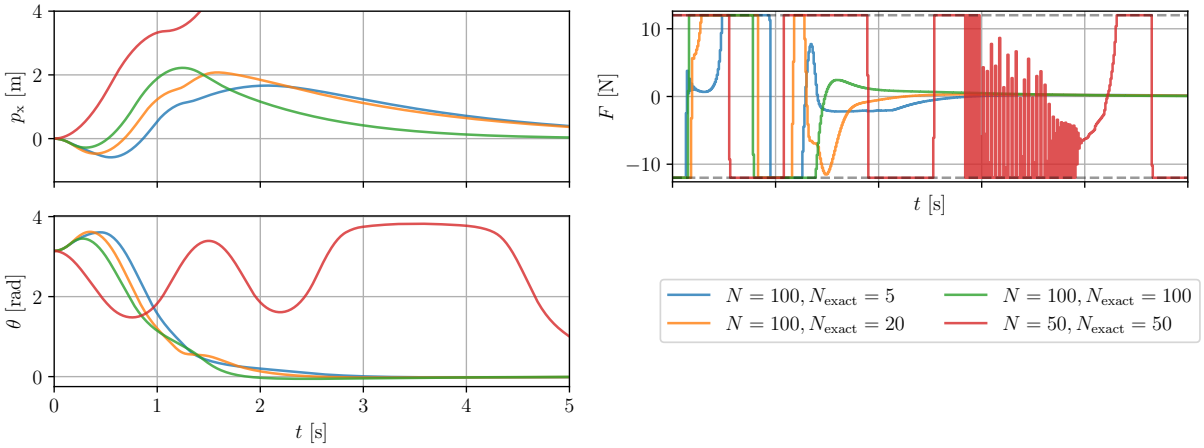

FIGURE 9 Closed-loop trajectories corresponding to different controller variants with partial tightening. The total number of shooting intervals N and the ones handled exactly N_{exact} are varied. The shooting intervals are part of the tightened horizon.

with $Q = \text{diag}\left(0.1, 1, 0.1, 2 \cdot 10^{-3}\right), R = 5 \cdot 10^{-4}$. For the terminal cost, we use $x^\top P x$, where P is the solution of the continuous-time algebraic Riccati equation with cost and dynamics linearized at the steady-state. The only constraint is that the control input is $-12 \leq v \leq 12$. When using partial tightening, the constraint is replaced with an additional log-barrier term as in (10) corresponding to this constraint with $\tau = 5$ and a GGN Hessian approximation is used. While previous works only used custom

implementations of partial tightening, the MOCP interface allows a convenient formulation in established software and the source code to reproduce the results presented in this Section is publicly available[¶].

This benchmark compares controllers in a closed-loop simulation with initial state $x_0 = (0, \pi, 0, 0)$ for a duration of 5s with a time step of 0.01s. The controllers use different underlying OCP formulations, where the length of the overall horizon and the tightened horizon are varied. The tightened horizon is implemented as a second phase of an MOCP formulation. The length of a shooting interval is fixed to 0.01s and the dynamics are discretized using an RK4 integrator. We vary the total number of shooting intervals N and the number of shooting intervals on which the constraints are formulated exactly N_{exact}. The second phase contains $N - N_{\text{exact}}$ shooting intervals, on which the control bounds are replaced with logarithmic barriers.

All controllers use the real-time iteration algorithm. The QPs are solved using partial condensing, such that the blocks corresponding to the tightened horizon are condensed into one block and the remaining blocks remain uncondensed. In order to perform as many operations as possible in the preparation phase, `acados` internally implements functions that assume that only matrices of the QP are known and a second one that completes the computations once the vector quantities are known. For the `acados` module that performs the condensing and QP solution has a split functionality, namely `condense_lhs()` and `condense_rhs_and_solve()`.

Table 2 gives an overview on the controllers closed-loop performance and the computation times split into preparation and feedback phase. Figure 9 visualizes the closed-loop trajectories for different controller variants. Figure 8 and Table 2 show that the preparation time is consistent between all solver variants with $N = 100$ and roughly halved for the variant with $N = 50$. While the controller with $N = 50, N_{\text{exact}} = 50$ fails at swinging up the pendulum, the other controllers succeed at this task. Comparing the variants with $N = 100$, one can see that decreasing N_{exact} results in an increase in relative suboptimality and a decrease in associated computational complexity, more specifically a decrease int he computation time of the feedback phase. Overall, the results are very similar to the ones reported in the original benchmark[61]. Note that in our implementation, the barrier parameter τ could be easily increased over the shooting intervals, resulting in a progressive tightening formulation that is not just partial tightening. In summary, partial and progressive tightening formulations allow one to trade-off computational complexity and closed-loop performance and can be formulated conveniently as MOCPs.

6 | CONCLUSION

This paper gives an overview on multi-phase OCP (MOCP) formulations and their efficient treatment using multiple shooting. Several approaches are presented which allow to formulate a continuous-time OCP in a successively approximate way which require an MOCP formulation. The work provides an overview on different control parametrizations for use within multiple shooting, such as piecewise polynomial controls of different degrees and closed-loop costing variants, and motivated their use. These control parametrizations have been compared on a benchmark example from previous work. Moreover, we demonstrate the efficiency of NMPC controllers based on an MOCP formulation with an approximate model in a second phase. Lastly, we show how partial and progressive tightening OCPs can be phrased as MOCPs. These examples show that the added degrees of freedom allow one to develop NMPC controllers, which outperform ones limited to single-phase OCP formulations. While other state-of-the-art software packages are limited to single-phase OCP formulations or use general purpose NLP solvers instead of structure-exploiting algorithms tailored to OCPs, the new `acados` feature allows for both a convenient formulation and the generation of efficient solver for MOCPs. We believe that this new feature will make efficient solvers for MOCP formulations more available to NMPC practitioners and spread their use in real-world applications.

ACKNOWLEDGMENTS

The authors want to thank Rudolf Reiter for fruitful discussions.

FINANCIAL DISCLOSURE

This research was supported by DFG via Research Unit FOR 2401 and projects 424107692, 504452366, by BMWK 03EN3054B, and by the EU via ELO-X 953348.

CONFLICT OF INTEREST

The authors declare no potential conflict of interests.

[¶] https://github.com/FreyJo/ocp_solver_benchmark/blob/main/experiments/zanelli_partial_tightening.py

REFERENCES

1. Dabbene F, Cannon M, Chamanbaz M, Isaksson A, Mammarella M, Raimondo D. Guest Editorial Special Issue on State-of-the-Art Applications of Model Predictive Control. *IEEE Transactions on Control Systems Technology.* 2023;31(5):1965–1970.
2. Hänggi S, Frey J, Dooren vS, Diehl M, Onder CH. A Modular Approach for Diesel Engine Air Path Control Based on Nonlinear MPC. *IEEE Transactions on Control Systems Technology.* 2022:1-16. doi: 10.1109/TCST.2022.3228203
3. Zanelli A, Kullick J, Eldeeb H, Frison G, Hackl C, Diehl M. Continuous Control Set Nonlinear Model Predictive Control of Reluctance Synchronous Machines. *IEEE Transactions on Control Systems Technology.* 2021:1-12. doi: 10.1109/TCST.2020.3043956
4. Carlos BB, Sartor T, Zanelli A, Diehl M, Oriolo G. Least Conservative Linearized Constraint Formulation for Real-Rime Motion Generation. In: 2020:9519–9525.
5. Carlos BB, Sartor T, Zanelli A, et al. An Efficient Real-Time NMPC for Quadrotor Position Control under Communication Time-Delay. In: 2020:982-989
6. Gao Y, Messerer F, Frey J, van Duijkeren N, Diehl M. Collision-free Motion Planning for Mobile Robots by Zero-order Robust Optimization-based MPC. In: 2023.
7. Norouzi A, Shahpouri S, Gordon D, et al. Deep learning based model predictive control for compression ignition engines. *Control Engineering Practice.* 2022;127:105299.
8. Romero A, Penicka R, Scaramuzza D. Time-optimal online replanning for agile quadrotor flight. *IEEE Robotics and Automation Letters.* 2022;7(3):7730–7737.
9. Frey J, Baumgärtner K, Diehl M. Gauss-Newton Runge-Kutta Integration for Efficient Discretization of Optimal Control Problems with Long Horizons and Least-Squares Costs. In: .
10. Vassiliadis V, Sargent R, Pantelides C. Solution of a class of multistage dynamic optimization problems. 1. Problems without path constraints. *Industrial and Engineering Chemistry Research.* 1994;10(33):2111–2122.
11. Magni L, Scattolini R. Stabilizing model predictive control of nonlinear continuous systems. *Annual Reviews in Control.* 2004;28:1–11.
12. Verschueren R, Frison G, Kouzoupis D, et al. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation.* 2021:147–183. doi: 10.1007/s12532-021-00208-8
13. Kouzoupis D. *Structure-exploiting numerical methods for tree-sparse optimal control problems*. PhD thesis. University of Freiburg, 2019.
14. Fiedler F, Karg B, Lüken L, et al. do-mpc: Towards FAIR nonlinear and robust model predictive control. *Control Engineering Practice.* 2023;140:105676.
15. Lucia S. *Robust Multi-stage Nonlinear Model Predictive Control*. PhD thesis. TU Dortmund, 2014.
16.
17. Patterson MA, Rao AV. GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming. *ACM Trans. Math. Softw.*. 2014;41(1). doi: 10.1145/2558904
18. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming.* 2006;106(1):25–57.
19. Gill P, Murray W, Saunders M. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Review.* 2005;47(1):99-131. doi: 10.1137/S0036144504446096
20. Ye H, Liu R. A multiphase optimal control method for multi-train control and scheduling on railway lines. *Transportation Research Part B: Methodological.* 2016;93:377-393. doi: https://doi.org/10.1016/j.trb.2016.08.002
21. Leineweber DB, Bauer I, Bock HG, Schlöder JP. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: theoretical aspects. *Computers and Chemical Engineering.* 2003;27:157–166.
22. Leineweber DB, Schäfer AAS, Bock HG, Schlöder JP. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications. *Computers and Chemical Engineering.* 2003;27:167–174.
23. Frison G, Kouzoupis D, Sartor T, Zanelli A, Diehl M. BLASFEO: Basic Linear Algebra Subroutines For Embedded Optimization. *ACM Transactions on Mathematical Software (TOMS).* 2018;44(4):42:1–42:30.
24. Diehl M. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis. University of Heidelberg, 2001.
25. Frey J, Nurkanović A, Diehl M. Advanced-Step Real-Time Iterations With Four Levels – New Error Bounds and Fast Implementation in acados. *IEEE Control Systems Letters.* 2024. doi: 10.1109/LCSYS.2024.3412007
26. Mayne D. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *Int. J. Control.* 1966;3(1):85–96.
27. Kiessling D, Baumgärtner K, Frey J, Decré W, Swevers J, Diehl M. Fast Generation of Feasible Trajectories in Direct Optimal Control. *IEEE Control Systems Letters.* 2024.
28. Kouzoupis D, Frison G, Zanelli A, Diehl M. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics.* 2018;46(4):863–882.
29. Vanroye L, Sathya A, De Schutter J, Decré W. Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control. In: IEEE. 2023:10036–10043.
30. Frison G, Diehl M. HPIPM: a high-performance quadratic programming framework for model predictive control. In: 2020; Berlin, Germany.
31. Ferreau HJ, Kirches C, Potschka A, Bock HG, Diehl M. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation.* 2014;6(4):327–363.
32. Arnstrom D, Bemporad A, Axehill D. A Dual Active-Set Solver for Embedded Quadratic Programming Using Recursive LDL^T Updates. *IEEE Transactions on Automatic Control.* 2022. doi: 10.1109/TAC.2022.3176430
33. Stellato B, Geyer T, Goulart PJ. High-Speed Finite Control Set Model Predictive Control for Power Electronics. *IEEE Trans. Automat. Control.* 2017;32(5):4007 – 4020.
34. Frasch JV, Vukov M, Ferreau H, Diehl M. A dual Newton strategy for the efficient solution of sparse quadratic programs arising in SQP-based nonlinear MPC. 2013. Optimization Online 3972.
35. Frison G, Kouzoupis D, Jørgensen JB, Diehl M. An Efficient Implementation of Partial Condensing for Nonlinear Model Predictive Control. In: 2016:4457–4462.
36. Axehill D. Controlling the level of sparsity in MPC. *Systems & Control Letters.* 2015;76:1–7.
37. Frey J, De Schutter J, Diehl M. Fast integrators with sensitivity propagation for use in CasADi. In: 2023.

38. Andersson JAE, Gillis J, Horn G, Rawlings JB, Diehl M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation.* 2019;11(1):1–36. doi: 10.1007/s12532-018-0139-4
39. Gillis J, Vandewal B, Pipeleers G, Swevers J. Effortless modeling of optimal control problems with rockit. In: 2020.
40. Sopasakis P, Fresk E, Patrinos P. OpEn: Code Generation for Embedded Nonconvex Optimization. In: .
41. Englert T, Völz A, Mesmer F, Rhein S, Graichen K. A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC). *Optimization and Engineering.* 2019;20(3):769–809. doi: 10.1007/s11081-018-9417-2
42. Zanelli A, Domahidi A, Jerez JL, Morari M. FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control.* 2017:13–29. doi: 10.1080/00207179.2017.1316017
43. Domahidi A, Zgraggen A, Zeilinger MN, Morari M, Jones CN. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In: 2012; Maui, HI, USA:668–674.
44. Leineweber DB, Bauer I, Schäfer AAS, Bock HG, Schlöder JP. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. (Parts I and II). *Computers and Chemical Engineering.* 2003;27:157–174.
45. Schultz G, Mombaur K. Modeling and optimal control of human-like running. *IEEE/ASME Transactions on mechatronics.* 2009;15(5):783–792.
46. Diehl M, Schäfer A, Bock HG, Schlöder JP. Optimization of Multiple-Fraction Batch Distillation with Recycled Waste Cuts. *AIChE Journal.* 2002;48(12):2869–2874. doi: 10.1002/aic.690481214
47. Bock HG, Plitt KJ. A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems. In: Pergamon Press 1984:242–247.
48. Frey J, Cairano SD, Quirynen R. Active-Set based Inexact Interior Point QP Solver for Model Predictive Control. In: 2020.
49. Schlipf D, Schlipf DJ, Kühn M. Nonlinear model predictive control of wind turbines using LIDAR. *Wind energy.* 2013;16(7):1107–1129.
50. De Nicolao G, Magni L, Scattolini R. Stabilizing Receding-Horizon control of nonlinear time varying systems. *IEEE Transactions on Automatic Control.* 1998;AC-43(7):1030–1036.
51. De Nicolao G, Magni L, Scattolini R. Stabilizing nonlinear receding horizon control via a nonquadratic terminal state penalty. In: 1996; Lille:185–187.
52. Diehl M, Magni L, Nicolao GD. Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control.* 2004;28(1):37–45. doi: 10.1016/j.arcontrol.2004.01.011
53. Bertsekas D, Tsitsiklis J. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
54. Magni L, De Nicolao G, Magnani L, Scattolini R. A stabilizing model-based predictive control for nonlinear systems. *Automatica.* 2001;37(9):1351–1362.
55. Quirynen R, Vukov M, Zanon M, Diehl M. Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. *Optimal Control Applications and Methods.* 2014;36:685–704.
56. Rawlings JB, Mayne DQ, Diehl MM. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill. 2nd ed., 2017.
57. Verschueren R, Duijkeren vN, Quirynen R, Diehl M. Exploiting Convexity in Direct Optimal Control: a Sequential Convex Quadratic Programming Method. In: 2016.
58. Baumgärtner K, Diehl M. The Extended Gauss-Newton Method for Nonconvex Loss Functions and its Application to Time-Optimal Model Predictive Control. In: 2022.
59. Baumgärtner K, Wang Y, Zanelli A, Diehl M. Fast Nonlinear Model Predictive Control using Barrier Formulations and Squashing with a Generalized Gauss-Newton Hessian. In: IEEE. 2022:558–563.
60. Baumgärtner K, Zanelli A, Diehl M. Stability Analysis of Nonlinear Model Predictive Control with Progressive Tightening of Stage Costs and Constraints. *IEEE Control Systems Letters.* 2023.
61. Zanelli A, Quirynen R, Frison G, Diehl M. A Partially Tightened Real-Time Iteration Scheme for Nonlinear Model Predictive Control. In: 2017; Melbourne, Australia.
62. Zanelli A. *Inexact methods for nonlinear model predictive control: stability, applications, and software*. PhD thesis. University of Freiburg, 2021.
63. Reiter R, Baumgärtner K, Quirynen R, Diehl M. Progressive Smoothing for Motion Planning in Real-Time NMPC. *Proceedings of the European Control Conference (ECC).* 2024.
64. Zanelli A, Horn G, Frison G, Diehl M. Nonlinear Model Predictive Control of a Human-sized Quadrotor. In: 2018:1542–1547.
65. Valverde G, Van Cutsem T. Model predictive control of voltages in active distribution networks. *IEEE Transactions on Smart Grid.* 2013;4(4):2152–2161.
66. Frison G. *Algorithms and Methods for High-Performance Model Predictive Control*. PhD thesis. Technical University of Denmark (DTU), 2015.
67. Dhaouadi R, Hatab AA. Dynamic modelling of differential-drive mobile robots using Lagrange and Newton-Euler methodologies: A unified framework. *Advances in Robotics & Automation.* 2013;2(2):1–7.