International Journal of Control

ISSN: 0020-7179 (Print) 1366-5820 (Online) Journal homepage: www.tandfonline.com/journals/tcon20

Structure-exploiting Newton-type method for optimal control of switched systems

Sotaro Katayama & Toshiyuki Ohtsuka

To cite this article: Sotaro Katayama & Toshiyuki Ohtsuka (2024) Structure-exploiting Newton-type method for optimal control of switched systems, International Journal of Control, 97:8, 1717-1733, DOI: 10.1080/00207179.2023.2227285

To link to this article: https://doi.org/10.1080/00207179.2023.2227285

Published online: 27 Jun 2023.

INTERNATIONAL JOURNAL OF CONTROL
2024, VOL. 97, NO. 8, 1717–1733
https://doi.org/10.1080/00207179.2023.2227285

Taylor & Francis
Taylor & Francis Group

Structure-exploiting Newton-type method for optimal control of switched systems

Sotaro Katayama and Toshiyuki Ohtsuka

Graduate School of Informatics, Kyoto University, Kyoto, Japan

ABSTRACT

This study proposes an efficient Newton-type method for the optimal control of switched systems under a given mode sequence. A mesh-refinement-based approach is utilised to discretise continuous-time optimal control problems (OCPs) and formulate a nonlinear program (NLP), which guarantees the local convergence of a Newton-type method. A dedicated structure-exploiting algorithm (Riccati recursion) is proposed to perform a Newton-type method for the NLP efficiently because its sparsity structure is different from a standard OCP. The proposed method computes each Newton step with linear time-complexity for the total number of discretization grids as the standard Riccati recursion algorithm. Additionally, the computation is always successful if the solution is sufficiently close to a local minimum. Conversely, general quadratic programming (QP) solvers cannot accomplish this because the Hessian matrix is inherently indefinite. Moreover, a modification on the reduced Hessian matrix is proposed using the nature of the Riccati recursion algorithm as the dynamic programming for a QP subproblem to enhance the convergence. A numerical comparison is conducted with off-the-shelf NLP solvers, which demonstrates that the proposed method is up to two orders of magnitude faster. Whole-body optimal control of quadrupedal gaits is also demonstrated and shows that the proposed method can achieve the whole-body model predictive control (MPC) of robotic systems with rigid contacts.

ARTICLE HISTORY
Received 20 December 2021
Accepted 7 June 2023

KEYWORDS
Optimal control; switched systems; numerical optimization; model predictive control: legged robots

1. Introduction

Switched systems are a class of hybrid systems consisting of a finite number of subsystems and switching laws of active subsystems. Many practical control systems are modeled as switched systems, such as real-world complicated process systems (Bürger et al., 2019), automotive systems with gear shifts (Robuschi et al., 2021), and robotic systems with rigid contacts (Farshidian et al., 2017; Li & Wensing, 2020). It is difficult to solve the optimal control problems (OCPs) of such systems because these OCPs generally involve mixed-integer nonlinear programs (MINLPs) (Belotti et al., 2013). Therefore, model predictive control (MPC) (Rawlings et al., 2017), in which OCPs must be solved in real-time, is particularly difficult for switched systems. One of the most practical approaches for solving the OCPs of switched systems is the combinatorial integral approximation (CIA) decomposition method (Bürger et al., 2019; Sager et al., 2011). CIA decomposition relaxes the MINLP into a nonlinear program (NLP) in which the binary variables are relaxed into continuous variables. Subsequently, the integer variables are approximately reconstructed from the relaxed NLP solution by solving a mixed-integer linear program. However, vanishing constraints, which typically model mode-dependent path constraints, raise numerical issues due to the violation of the linear independence constraint qualification (LICQ) (Jung et al., 2013). A similar smoothing approach is used for discrete natures when solving OCPs for mechanical systems with rigid contacts, which are often approximately formulated as mathematical programs with complementarity constraints (MPCCs) (Posa et al., 2014; Yunt, 2011). However, this case has the same LICQ problem as the vanishing constraints, requiring a significant amount of computational time to alleviate the numerical ill-conditioning. Furthermore, it often suffers from undesirable stationary points (Nurkanović et al., 2020).

Another tractable and practical approach for the optimal control of switched systems is to fix the switching (mode) sequence. For example, in robotics applications, a high-level planner computes the feasible contact sequence taking perception into account. Subsequently, the sequence is provided to a lower-level optimal control-based dynamic motion planner or MPC controller (Grandia et al., 2021; Jenelten et al., 2020; Kuindersma et al., 2016). The lower-level dynamic planner or MPC then discovers the optimal switching times and optimal control input. An advantage of this approach over the CIA decomposition and MPCC approaches is that the optimization problem is smooth and can, therefore, be efficiently solved without suffering from the LICQ problem. For example, Johnson and Murphey (2011) and Stellato et al. (2017) proposed efficient Newton-type methods for OCPs of switched systems with autonomous subsystems. That is, the switched systems only included a switching signal but no continuous control input, which is called the switching-time optimization (STO) problem. The STO approach can be more efficient than the CIA decomposition because the STO problem can be formulated as a smooth and tractable NLP, as numerically shown in Stellato et al. (2017). However, once the switched system includes a continuous control input, the efficient methods of Johnson and Murphey (2011)

CONTACT Sotaro Katayama katayama.25w@st.kyoto-u.ac.jp

and Stellato et al. (2017) cannot be applied. To the best of our knowledge, there is no efficient numerical method for OCPs of such systems. As a result, many real-world robotic applications are limited to focusing on dynamic motion planning with fixed switching instants (Di Carlo et al., 2018; Farshidian et al., 2017; Mastalli et al., 2020).

The two-stage approach has been studied for the OCPs of switched systems with continuous control input (Farshidian et al., 2017; Fu & Zhang, 2021; Li & Wensing, 2020; Xu & Antsaklis, 2002, 2004). A general two-stage approach was proposed in Xu and Antsaklis (2002, 2004). In this approach, the optimization problem to determine the control input and switching instants was decomposed into an STO problem with a fixed control input (upper-level problem) and standard OCP that only determined the control input with fixed switching instants (lower-level problem). In this framework, off-the-shelf OCP solvers can be used for a lower-level problem. Xu and Antsaklis (2004) formulated the lower-level continuous-time OCP as a two-point boundary-value problem and leveraged off-the-shelf shooting methods to solve it. Farshidian et al. (2017), Li and Wensing (2020), and Xu and Antsaklis (2002) formulated the lower-level problem as an NLP by using the direct single shooting method and solved it using off-the-shelf Newton-type methods. However, these studies still lack convergence speed because each of these two stages does not take the other stage into account when solving its own optimization problem. As a result, the application examples of Farshidian et al. (2017) and Li and Wensing (2020) were limited to off-line computation for the trajectory optimization problems of simplified robot models. Moreover, they could not guarantee the local convergence or address inequality constraints. The study of Fu and Zhang (2021) was the first study that guaranteed convergence with a finite number of iterations and treated inequality constraints within a two-stage framework. However, it still required extensive computational time until convergence, even for a considerably simple linear quadratic example.

Other approaches simultaneously optimise the switching instants and other variables, such as the state and control input (Betts, 2010; Katayama et al., 2020; Katayama & Ohtsuka, 2021c; Patterson & Rao, 2014). A multi-phase trajectory optimization (Betts, 2010; Patterson & Rao, 2014) naturally incorporated the STO problem into the direct transcription, solving the NLP to simultaneously determine all variables (including the state, control input, and switching instants) using general-purpose off-the-shelf NLP solvers, such as Ipopt (Wächter & Biegler, 2006). However, the computational speed of general-purpose linear solvers used in off-the-shelf NLPs can typically be further improved, particularly, for large-scale systems because they have a certain sparsity structure (Rao et al., 1998; Wang & Boyd, 2010; Zanelli et al., 2020). In Katayama et al. (2020), we applied the simultaneous approach with the direct single-shooting method and achieved real-time MPC for a simple walking robot using the Newton–Krylov type method (Ohtsuka, 2004). In Katayama and Ohtsuka (2021c), we formulated an NLP using the direct multiple-shooting method (Bock & Plitt, 1984) and proposed a Riccati recursion algorithm for the NLP, which achieved faster computational time compared with the two-stage methods. However, these methods lacked the convergence guarantee due to the irregular discretization of the continuous-time OCP, which changed the problem structure throughout the Newton-type iterations. As a result, the method from Katayama and Ohtsuka (2021c) could only converge when the initial guess of the switching instants was close to the optimal one in the numerical example.

This study proposes an efficient Newton-type method for optimal control of switched systems under a given mode sequence. First, a mesh-refinement-based approach is proposed to discretise the continuous-time OCP using the direct multiple-shooting method (Bock & Plitt, 1984) to formulate an NLP that facilitates the local convergence of the Newton-type methods. Second, a dedicated efficient structure-exploiting algorithm (Riccati recursion algorithm Rao et al., 1998) is proposed for the Newton-type method because the sparsity structure of the NLP is different from the standard OCP. The proposed method computes each Newton step with linear time-complexity of the total number of the discretization grids as the standard Riccati recursion algorithm. Additionally, it can always solve the Karush–Kuhn–Tucker (KKT) systems arising in the Newton-type method if the solution is sufficiently close to a local minimum, so that the second-order sufficient condition (SOSC) holds. This is in contrast to some general quadratic programming (QP) solvers that cannot treat the proposed formulation because the Hessian matrix is inherently indefinite. Third, a modification on the reduced Hessian matrix is proposed to enhance the convergence using the nature of the Riccati recursion algorithm as the dynamic programming (Bertsekas, 2005) for a QP subproblem. Two numerical experiments were conducted to demonstrate the efficiency of the proposed method: a comparison with off-the-shelf NLP solvers and testing the whole-body optimal control of quadrupedal gaits. The comparison with off-the-shelf NLP solvers showed that the proposed method could solve the OCPs that the sequential quadratic programming (SQP) method with qpOASES (Ferreau et al., 2014) or OSQP (Stellato et al., 2020) failed to solve and was up to two orders of magnitude faster than a general NLP solver, Ipopt (Wächter & Biegler, 2006). The whole-body optimal control of quadrupedal gaits showed that the proposed method achieved the whole-body MPC of robotic systems with rigid contacts.

The remainder of this paper is organised as follows. A mesh-refinement-based discretization method of the continuous-time OCP is presented in Section 2. The KKT system to be solved to compute the Newton-step is discussed in Section 3, and the Riccati recursion algorithm to solve the KKT system and its convergence properties are described in Section 4. The reduced Hessian modification using the Riccati recursion algorithm is also described in this section. The above formulations and Riccati recursion algorithm are extended to switched systems with state jumps and switching conditions in Section 5, representing robotic systems with contacts. A numerical comparison of the proposed method with off-the-shelf NLP solvers and examples of whole-body optimal control of a quadrupedal robot are presented in Section 6. Finally, a brief summary and mention of future work are presented in Section 7.

Notation and preliminaries: The Jacobians and Hessians of a differentiable function using certain vectors are described as follows: $\nabla_x f(x)$ denotes $(\frac{\partial f}{\partial x})^\mathrm{T}(x)$, and $\nabla_{xy} g(x, y)$ denotes $\frac{\partial^2 g}{\partial x \partial y}(x, y)$. A diagonal matrix whose elements are a vector x is

denoted as diag(x). A vector of an appropriate size with all components presented by $\alpha \in \mathbb{R}$ is denoted as $\alpha 1$. All functions are assumed to be twice-differentiable.

2. Problem formulation

We consider a switched system consisting of $K+1$ ($K > 0$) subsystems, which is expressed as

$$\dot{x}(t) = f_k(x(t), u(t)), \quad t \in [t_{k-1}, t_k],\ k \in \mathcal{K}. \tag{1}$$

The system is subject to the following constraints:

$$g_k(x(t), u(t)) \le 0, \quad t \in [t_{k-1}, t_k],\ k \in \mathcal{K}, \tag{2}$$

where $x(t) \in \mathbb{R}^{n_x}$ denotes the state, $u(t) \in \mathbb{R}^{n_u}$ denotes the control input, $f_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, and $g_k : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$. $\mathcal{K} := \{1, \dots, K+1\}$ denotes the indices of the active subsystems. Note that the index k is also referred to as a 'phase' in this study. t_0 and t_{K+1} denote the fixed initial and terminal times of the horizon, respectively. $t_k, k \in \{1, \dots, K\}$ denotes the switching instant from phase k to phase $k+1$. We assume that the active subsystems switch in order from 1 to $K+1$ over the horizon $[t_0, t_f]$, that is, $t_0 \le t_1 \le \cdots \le t_K \le t_{K+1}$. We further assume that each subsystem $k \in \{0, \dots, K\}$ has to be active, at least for a given time duration $\underline{\Delta}_k \ge 0$, called the minimum dwell-time. The OCP of the switched system for a given initial state $x(t_0) \in \mathbb{R}^{n_x}$ is expressed as

$$\min_{u(\cdot), t_1, \dots, t_K} \quad J = V_f(x(t_{K+1})) + \sum_{k=1}^{K+1} \int_{t_{k-1}}^{t_k} l_k(x(\tau), u(\tau))\, \mathrm{d}\tau \tag{3a}$$

$$\text{s.t.} \quad (1),\ (2),$$

$$t_{k-1} + \underline{\Delta}_k \le t_k, \quad k \in \mathcal{K}. \tag{3b}$$

Next, a discretization method and mesh-refinement-based solution approach are proposed for the OCP to guarantee the local convergence. Figure 1 illustrates the proposed discretization method. The OCP is discretised using the direct multiple-shooting method (Bock & Plitt, 1984) based on the forward Euler method so that all the time-steps in phase $k \in \mathcal{K}$ are equal. Note that it is easy to extend the proposed method for higher-order explicit integration schemes to the direct multiple-shooting method, such as the fourth-order explicit Runge–Kutta method, as long as the Riccati recursion algorithm can be applied to the integration schemes. We introduce N grid points over the horizon, the discretised state $X := \{x_0, \dots, x_N\}$, discretised control input $U := \{u_0, \dots, u_{N-1}\}$, and switching instants $T := \{t_1, \dots, t_K\}$. The NLP is then expressed as

$$\min_{X,U,T} \quad J = V_f(x_N) + \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} l_k(x_i, u_i) \Delta\tau_k, \tag{4a}$$

$$\text{s.t.} \quad x_0 - \bar{x} = 0, \tag{4b}$$

$$x_i + f_k(x_i, u_i)\Delta\tau_k - x_{i+1} = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{4c}$$

$$g_k(x_i, u_i) \le 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{4d}$$

$$t_{k-1} + \underline{\Delta}_k - t_k \le 0, \quad k \in \mathcal{K}. \tag{4e}$$

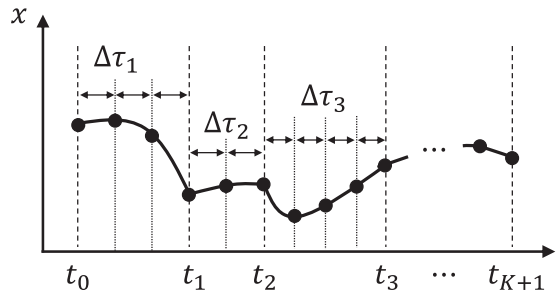

Figure 1. Proposed discretization method for the OCPs of switched systems.

Here, $\mathcal{I}_k$ is the set of stage indices at phase k (that is, the set of time stages where the subsystem equation $f_k(x_i, u_i)$ is active). Note that the last stage N is not included in any $\mathcal{I}_k$. That is, $0 \in \mathcal{I}_0$, $(\max \mathcal{I}_k) + 1 = \min \mathcal{I}_{k+1}$ for $k \in \{1, \dots, K\}$, and $N \notin \mathcal{I}_{K+1}$ hold for $\mathcal{I}_k$. $\Delta\tau_k$ is the time step at phase k, defined as

$$\Delta\tau_k := \frac{t_k - t_{k-1}}{N_k}, \quad k \in \mathcal{K}, \tag{4f}$$

where N_k is the number of grids in phase k, that is, $N_k = \max \mathcal{I}_k - \min \mathcal{I}_k$. Finally, the initial state condition is lifted as (4b), possibly for a real-time MPC implementation (Diehl et al., 2005).

The continuous-time OCP (3) is solved using an adaptive mesh-refinement approach (Betts, 2010), which consists of a solution of the NLP (4) using a Newton-type method and mesh-refinement due to the changes of $\Delta\tau_k$, as shown in Algorithm 1. After solving the NLP, the size of the discretization steps $\Delta\tau_k$ is checked for each $k \in \mathcal{K}$. If the step is too large, that is, if it exceeds the specified threshold $\Delta\tau_{\max}$, the solution at phase k may be not accurate. Therefore, the number of grids for phase k is increased. Conversely, if the step is too small ($\Delta\tau < \Delta\tau_{\min}$), the number of grids for phase k is reduced to decrease the computational time of the next NLP step. The algorithm terminates if some criteria (for example, l_2 norm of the KKT residual), which is denoted as 'NLP error' in Algorithm 1, is smaller than a predefined threshold $\epsilon > 0$ and the discretization steps pass the checks. By appropriately choosing $\Delta\tau_{\min}$, the NLP dimension is almost constant throughout Algorithm 1. This is an advantage of the proposed method over direct transcription methods, such as Patterson and Rao (2014), whose NLP dimension is unknown before being solved.

It is worth noting that the NLP (4) is smooth and its structure does not change within each NLP step. Therefore, the Newton-type method for the NLP (4) is always tractable.

Remark 2.1: It is trivial to show the local (superlinear) convergence of Newton-type methods for the NLP (4) under some reasonable assumptions (Nocedal & Wright, 2006). Moreover, if the solution guess after the mesh-refinement of Algorithm 1 is sufficiently close to a local minimum of the NLP after the mesh-refinement (that is, the mesh-refinement is sufficiently accurate), then the local convergence of the overall Algorithm 1 is also guaranteed. In contrast, previous methods cannot guarantee the local convergence, such as the two-stage approaches (Farshidian et al., 2017; Li & Wensing, 2020; Xu & Antsaklis, 2002, 2004) and simultaneous approaches with the other

Algorithm 1 Adaptive mesh-refinement approach for the optimal control of switched systems (3)

Require: The initial state $x(t_0)$, initial guess of the solution X, U, T, initial guess of the Lagrange multipliers, maximum and minimum discretization step sizes $\Delta\tau_{\max} > \Delta\tau_{\min} > 0$, and convergence tolerance $\epsilon > 0$.
Ensure: Optimal solution X, U, T.
1: **while** NLP error $> \epsilon$ **do**
2: **if** $\Delta\tau_k < \Delta\tau_{\min}$ **then**
3: Mesh-refinement for phase k (decrease grids).
4: **end if**
5: Solve the NLP (4) using the Newton-type method.
6: **if** $\Delta\tau_k > \Delta\tau_{\max}$ **then**
7: Mesh-refinement for phase k (increase grids).
8: **end if**
9: **end while**

discretization methods (Katayama et al., 2020; Katayama & Ohtsuka, 2021c).

Note that it is assumed throughout this study that the state Equation (1), inequality constraints (2), and cost function (3a) do not depend on the time. That is, they are time invariant for notational simplicity. However, it is possible to extend the proposed method to time-varying cases, where the time of each grid i that depends on the duration of phases $t_k - t_{k-1}$ in (1), (2), and (3a) must be taken into account. Then, additional sensitivities of (1), (2), and (3a) with respect to the switching times are introduced in the KKT conditions and systems, which are derived in the next section. The formulations and methods of this study can be directly applied to such cases as long as the structure of the KKT system has the same form as the next section.

3. KKT system for Newton-type method

Next, the KKT system is derived to compute the Newton step of the NLP (4). The inequality constraints are treated with the primal–dual interior point method (Nocedal & Wright, 2006; Wächter & Biegler, 2006). That is, the slack variables $z_0, \dots, z_{N-1} \in \mathbb{R}^{n_g}$ and $w_1, \dots, w_{K+1} \in \mathbb{R}$ are introduced for (4d) and (4e), respectively. The equality constraints are then considered, which are expressed as

$$r_{g,i} := g_k(x_i, u_i) + z_i = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{5a}$$

$$r_{\Delta,k} := t_{k-1} + \underline{\Delta}_k - t_k + w_k = 0, \quad k \in \mathcal{K}, \tag{5b}$$

instead of the inequality constraints (4d) and (4e). The barrier functions $-\epsilon \sum_{i=0}^{N-1} \ln z_i - \epsilon \sum_{k=1}^{K+1} \ln w_k$ are also added to the cost function (4a), where $\epsilon > 0$ is the barrier parameter. An NLP associated with the barrier parameter ϵ is then expressed as

$$\begin{aligned} \min_{X,U,T,Z,W} \quad & \tilde{J} = V_f(x_N) + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{I}_k} l_k(x_i, u_i)\Delta\tau_k \\ & - \epsilon \sum_{i=0}^{N-1} \ln z_i - \epsilon \sum_{k=1}^{K+1} \ln w_k \end{aligned} \tag{6a}$$

$$\text{s.t.} \quad \text{(4b)}, \ \text{(4c)}, \ \text{(5a)}, \ \text{(5b)}, \tag{6b}$$

where $Z := \{z_0, \dots, z_{N-1}\}$ and $W := \{w_1, \dots, w_{K+1}\}$.

We then derive the KKT conditions for the NLP (6). To this end, $\lambda_0, \dots, \lambda_N \in \mathbb{R}^{n_x}$ are introduced as the Lagrange multipliers with respect to (4b) and (4c); $\nu_0, \dots, \nu_{N-1} \in \mathbb{R}^{n_g}$ are with respect to (5a), and $\upsilon_1, \dots, \upsilon_{K+1} \in \mathbb{R}$ are with respect to (5b). The Lagrangian of the NLP (6) is then given by

$$\begin{aligned} \mathcal{L} := {} & V_f(x_N) + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{I}_k} H_k(x_i, u_i, \lambda_i)\Delta\tau_k \\ & + \sum_{i=0}^{N-1} \lambda_{i+1}^{\mathrm{T}}(x_i - x_{i+1}) + \lambda_0^{\mathrm{T}}(\bar{x} - x_0) \\ & - \epsilon \sum_{i=0}^{N-1} \ln z_i + \sum_{k\in\mathcal{K}} \sum_{i\in\mathcal{I}_k} \nu_i^{\mathrm{T}}(g_k(x_i, u_i) + z_i) \\ & - \epsilon \sum_{k=1}^{K+1} \ln w_k + \sum_{k=1}^{K+1} \upsilon_k(t_{k-1} + \underline{\Delta}_k - t_k + w_k), \end{aligned} \tag{7}$$

where

$$H_k(x_i, u_i, \lambda_{i+1}) := l_k(x_i, u_i) + \lambda_{i+1}^{\mathrm{T}} f_k(x_i, u_i) \tag{8}$$

is the Hamiltonian at phase $k \in \mathcal{K}$. The KKT conditions of the NLP (6a) are then composed by the equality constraints (4b), (4c), (5a), and (5b) and the first-order derivatives of the Lagrangian (7) with respect to X, U, T, Z, and W, which are expressed as

$$r_{x,N} := \nabla_x V_f(x_N) - \lambda_N = 0, \tag{9a}$$

$$\begin{aligned} r_{x,i} := {} & \nabla_x H_k(x_i, u_i, \lambda_{i+1})\Delta\tau_k \\ & + \nabla_x g_k^{\mathrm{T}}(x_i, u_i)\nu_i + \lambda_{i+1} - \lambda_i = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \end{aligned} \tag{9b}$$

$$\begin{aligned} r_{u,i} := {} & \nabla_u H_k(x_i, u_i, \lambda_{i+1})\Delta\tau_k \\ & + \nabla_u g_k^{\mathrm{T}}(x_i, u_i)\nu_i = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \end{aligned} \tag{9c}$$

$$r_{z,i} := \mathrm{diag}(z_i)\nu_i - \epsilon 1 = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{9d}$$

$$r_{w,k} := w_k \upsilon_k - \epsilon = 0, \quad k \in \mathcal{K}, \tag{9e}$$

and

$$\begin{aligned} & \frac{1}{N_k} \sum_{i\in\mathcal{I}_k} H_k(x_i, u_i, \lambda_{i+1}) \\ & \quad - \frac{1}{N_{k+1}} \sum_{i\in\mathcal{I}_{k+1}} H_{k+1}(x_i, u_i, \lambda_{i+1}) \\ & \quad - \upsilon_k + \upsilon_{k+1} = 0, \quad k \in \{1, \dots, K\}. \end{aligned} \tag{9f}$$

We refer to the above KKT conditions as the perturbed KKT conditions in the following because the complementary slackness regarding the inequality constraints are perturbed with ϵ as (9d) and (9e) (Nocedal & Wright, 2006). Next, the KKT system is derived to compute the Newton steps of all variables, that is, $\Delta x_0, \dots, \Delta x_N \in \mathbb{R}^{n_x}$, $\Delta u_0, \dots, \Delta u_{N-1}, \in \mathbb{R}^{n_u}$ $\Delta t_1, \dots, \Delta t_K \in \mathbb{R}$, and $\Delta\lambda_0, \dots, \Delta\lambda_N \in \mathbb{R}^{n_x}$. Note that the KKT system is herein considered as the standard primal–dual

interior point method (Nocedal & Wright, 2006; Wächter & Biegler, 2006), in which the Newton steps regarding the inequality constraints are eliminated (that is, $\Delta z_0, \dots, z_{N-1}$, $\Delta v_0, \dots, \Delta v_{N-1} \in \mathbb{R}^{n_g}$ and $\Delta w_1, \dots, \Delta w_{K+1}, \Delta \upsilon_1, \dots, \Delta \upsilon_{K+1} \in \mathbb{R}$). The KKT system of interest is then expressed as (note that $\Delta t_0 = \Delta t_{k+1} = 0$)

$$\Delta x_0 + x_0 - \bar{x} = 0, \tag{10a}$$

$$A_i \Delta x_i + B_i \Delta u_i + a_i(\Delta t_k - \Delta t_{k-1}) - \Delta x_{i+1} + \bar{x}_i = 0, i \in \mathcal{I}_k, \quad k \in \mathcal{K}, \tag{10b}$$

$$Q_{xx,N} \Delta x_N - \Delta \lambda_N + \bar{l}_{x,N} = 0, \tag{10c}$$

$$Q_{xx,i} \Delta x_i + Q_{xu,i} \Delta u_i + A_i^{\mathrm{T}} \Delta \lambda_{i+1} - \Delta \lambda_i + h_{x,i}(\Delta t_k - \Delta t_{k-1}) + \bar{l}_{x,i} = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{10d}$$

$$Q_{xu,i}^{\mathrm{T}} \Delta x_i + Q_{uu,i} \Delta u_i + B_i^{\mathrm{T}} \Delta \lambda_{i+1} + h_{u,i}(\Delta t_k - \Delta t_{k-1}) + \bar{l}_{u,i} = 0, \quad i \in \mathcal{I}_k,\ k \in \mathcal{K}, \tag{10e}$$

and

$$\begin{aligned} &\frac{1}{N_k} \sum_{i \in \mathcal{I}_k} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + a_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) \\ &\quad - \frac{1}{N_{k+1}} \sum_{i \in \mathcal{I}_{k+1}} \left(h_{x,i}^{\mathrm{T}} \Delta x_i + h_{u,i}^{\mathrm{T}} \Delta u_i + a_i^{\mathrm{T}} \Delta \lambda_{i+1} + \bar{h}_i \right) \\ &\quad + Q_{tt,k}(\Delta t_k - \Delta t_{k-1}) - Q_{tt,k+1}(\Delta t_{k+1} - \Delta t_k) \\ &\quad + \bar{q}_{t,k} - \bar{q}_{t,k+1} = 0, \quad k \in \{1, \dots, K\}, \end{aligned} \tag{10f}$$

where

$$\begin{aligned} Q_{xx,i} &:= \nabla_{xx} H_k(x_i, u_i, \lambda_{i+1}) \Delta \tau_k \\ &\quad + \nabla_x g_k^{\mathrm{T}}(x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(v_i) \nabla_x g_k^{\mathrm{T}}(x_i, u_i), \\ Q_{xu,i} &:= \nabla_{xu} H_k(x_i, u_i, \lambda_{i+1}) \Delta \tau_k \\ &\quad + \nabla_x g_k^{\mathrm{T}}(x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(v_i) \nabla_u g_k^{\mathrm{T}}(x_i, u_i), \\ Q_{uu,i} &:= \nabla_{uu} H_k(x_i, u_i, \lambda_{i+1}) \Delta \tau_k \\ &\quad + \nabla_u g_k^{\mathrm{T}}(x_i, u_i) \mathrm{diag}(z_i)^{-1} \mathrm{diag}(v_i) \nabla_u g_k^{\mathrm{T}}(x_i, u_i), \\ \bar{l}_{x,i} &:= r_{x,i} + \nabla_x g_k^{\mathrm{T}}(x_i, u_i) \mathrm{diag}(z_i)^{-1} (\mathrm{diag}(v_i) r_{g,i} - r_{z,i}), \\ \bar{l}_{u,i} &:= r_{u,i} + \nabla_u g_k^{\mathrm{T}}(x_i, u_i) \mathrm{diag}(z_i)^{-1} (\mathrm{diag}(v_i) r_{g,i} - r_{z,i}), \\ Q_{tt,k} &:= w_k^{-1} \upsilon_k, \quad \bar{q}_{t,k} := -\upsilon_k - w_k^{-1}(\upsilon_k r_{\Delta,k} - r_{w,k}), \\ A_i &:= I + \nabla_x f_k(x_i, u_i) \Delta \tau_k, \quad B_i := \nabla_x f_k(x_i, u_i) \Delta \tau_k, \\ a_i &:= \frac{1}{N_k} f_k(x_i, u_i), \quad h_i := \frac{1}{N_k} H_k(x_i, u_i, \lambda_{i+1}), \end{aligned}$$

and

$$h_{x,i} := \frac{1}{N_k} \nabla_x H_k(x_i, u_i, \lambda_{i+1}), \quad h_{u,i} := \frac{1}{N_k} \nabla_u H_k(x_i, u_i, \lambda_{i+1}).$$

In addition, $\bar{x}_i$ is residual in (4c). Note that some of the phase index k is omitted from the KKT system Equation (10) because the phase index k is determined from i, and the matrices and vectors (other than the Newton steps in (10)) are fixed once they are computed and therefore do not depend on the phase index k to solve the KKT system. The equations of the KKT system (10a)–(10f) correspond to the KKT conditions (4b), (4c), (9a)–(9c), and (9f), respectively.

Note that the KKT system (10) is equivalent to the KKT conditions of a QP subproblem

$$\begin{aligned} \min_{\substack{\Delta u_0, \dots, \Delta u_{N-1} \\ \Delta t_1, \dots, \Delta t_{K+1}}} \quad & \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k, i \neq N} \frac{1}{2} \left\{ \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix}^{\mathrm{T}} \right. \\ & \times \begin{bmatrix} 0 & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{xu,i} \\ h_{u,i} & Q_{ux,i} & Q_{uu,i} \end{bmatrix} \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \\ & \left. + \begin{bmatrix} \bar{h}_i \\ \bar{l}_{x,i} \\ \bar{l}_{u,i} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \right\} \\ & + \frac{1}{2} \Delta x_N^{\mathrm{T}} Q_{xx,N} \Delta x_N + \bar{l}_{x,N}^{\mathrm{T}} \Delta x_N \\ & + \sum_{k \in \mathcal{K}} \left\{ \frac{1}{2} Q_{tt,k} (\Delta t_k - \Delta t_{k-1})^2 \right. \\ & \left. + \bar{q}_{t,k} (\Delta t_k - \Delta t_{k-1}) \right\} \\ \text{s.t.} \quad & (10a),\ (10b), \end{aligned} \tag{11}$$

which is a quadratic approximation of the NLP (4). Subsequently, $\Delta \lambda_0, \dots, \Delta \lambda_N$ can be regarded as the Lagrange multipliers of the QP with respect to (10a) and (10b) (Nocedal & Wright, 2006).

Remark 3.1: The Hessian matrix of (11) is inherently indefinite, which makes solving the KKT system (10) difficult when using off-the-shelf QP solvers because they typically require a positive definite Hessian matrix. This can be explained with a block diagonal of the Hessian matrix (11), expressed as

$$\begin{bmatrix} 0 & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{uu,i} \\ h_{u,i} & Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix}. \tag{12}$$

This is indefinite due to the off-diagonal terms $h_{x,i}$ and $h_{u,i}$, even when: $\begin{bmatrix} Q_{xx,i} & Q_{uu,i} \\ Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix} \succ O$.

4. Riccati recursion to solve KKT systems

In this section, a Riccati recursion algorithm is presented to compute the Newton step of the NLP (4) by solving the KKT system (10). The sparsity structure of the KKT system (10) is no longer the same as the standard OCP, which prevents applying the off-the-shelf efficient Newton-type algorithms for OCPs (Rao et al., 1998; Wang & Boyd, 2010; Zanelli et al., 2020). Moreover, as stated in Remark 3.1, the Hessian matrix of the KKT system is indefinite and unsolvable using general QP solvers. Motivated by these problems, we propose a Riccati recursion algorithm that efficiently solves the KKT system, specifically with $O(N)$ computational time, whenever the SOSC holds. Moreover, a reduced Hessian modification method is

proposed to enhance the convergence when the SOSC does not hold, that is, when the reduced Hessian matrix is indefinite. As the standard Riccati recursion (Rao et al., 1998), the proposed method is composed of backward and forward recursions, which recursively eliminate the variables from the KKT system (10) backward in time and recursively compute the Newton step forward in time, respectively.

4.1 Backward recursion

In the backward recursion, the Newton steps are recursively eliminated from stages N to 0. Specifically, expressions of Δx_{i+1}, Δu_i, and $\Delta\lambda_i$ are derived at each stage $i \in I_k$ for Δx_i, Δt_k, and Δt_{k-1}, respectively, from (10b)–(10e). Moreover, variables are recursively eliminated from (10f) using these expressions. When the stage of interest changes from k to $k-1$, Δt_k is further eliminated from (10f). That is, an expression of Δt_k is derived with respect to Δx_{i_k} and Δt_{k-1}, where $i_k := \min \mathcal{I}_k$. This can be seen as the extension of the standard Riccati recursion that also recursively derives expressions of Δx_{i+1}, Δu_i, and $\Delta\lambda_i$ with respect to Δx_i (Rao et al., 1998; Rawlings et al., 2017).

4.1.1 Terminal stage

From (10c) at stage N, we obtain:

$$\Delta\lambda_N = P_N \Delta x_N - s_N, \tag{13a}$$

where

$$P_N = Q_{xx,N}, \quad s_N = -\bar{l}_{x,N}. \tag{13b}$$

4.1.2 Intermediate stages

Consider $i, i+1 \in \mathcal{I}_k$, $k \in \mathcal{K}$. Suppose that we have the expression of $\Delta\lambda_{i+1}$ with respect to Δx_{i+1}, Δt_k, and Δt_{k-1} as

$$\begin{aligned}\Delta\lambda_{i+1} = P_{i+1}\Delta x_{i+1} - s_{i+1} + \Psi_{i+1}(\Delta t_k - \Delta t_{k-1}) \\ + \Phi_{i+1}(-\Delta t_k),\end{aligned} \tag{14a}$$

where $P_{i+1} \in \mathbb{R}^{n_x \times n_x}$ and s_{i+1}, Ψ_{i+1}, $\Phi_{i+1} \in \mathbb{R}^{n_x}$. Furthermore, suppose that we have Equation (10f) for k and $k-1$ in which Δx_j for $j \geq i+2$, Δu_j for $j \geq i+1$, and $\Delta\lambda_j$ for $j \geq i+1$ are eliminated, that is, Equation (10f) for k and $k-1$ are reduced to

$$\begin{aligned}&\sum_{j\in\mathcal{I}_k,\, j\leq i} \left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + a_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) \\ &\quad + \Psi_{i+1}^{\mathrm{T}}\Delta x_{i+1} + \xi_{i+1}(\Delta t_k - \Delta t_{k-1}) + \chi_{i+1}(-\Delta t_k) + \eta_{i+1} \\ &\quad - \Phi_{i+1}^{\mathrm{T}}\Delta x_{i+1} - \chi_{i+1}(\Delta t_k - \Delta t_{k-1}) \\ &\quad - \rho_{i+1}(-\Delta t_k) - \iota_{i+1} = 0,\end{aligned} \tag{14b}$$

and

$$\begin{aligned}&\sum_{j\in\mathcal{I}_{k-1}} \left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + a_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) \\ &\quad - \sum_{j\in\mathcal{I}_k,\, j\leq i} \left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + a_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) \\ &\quad + Q_{tt,k-1}(\Delta t_{k-1} - \Delta t_{k-2}) + \bar{q}_{t,k-1} \\ &\quad - \Psi_{i+1}^{\mathrm{T}}\Delta x_{i+1} - \xi_{i+1}(\Delta t_k - \Delta t_{k-1}) \\ &\quad - \chi_{i+1}(-\Delta t_k) - \eta_{i+1} = 0,\end{aligned} \tag{14c}$$

where $\xi_{i+1}, \chi_{i+1}, \rho_{i+1}, \eta_{i+1}, \iota_{i+1} \in \mathbb{R}$. Note that $\Psi_{i+1} = 0$, $\Phi_{i+1} = 0$, $\xi_{i+1} = Q_{tt,K+1}$, $\chi_{i+1} = 0$, $\rho_{i+1} = 0$, $\eta_{i+1} = \bar{q}_{t,k+1}$, and $\iota_{i+1} = 0$ at stage $i = N-1$. Moreover, $\Psi_{i+1} = 0$, $\xi_{i+1} = Q_{tt,k+1}$, $\chi_{i+1} = 0$, and $\eta_{i+1} = \bar{q}_{t,k+1}$ at stages $i = \min \mathcal{I}_k - 1$ and $k \in \{2, \ldots, K\}$, as explained in 4.1.3. First, the following equations are introduced:

$$F_i := Q_{xx,i} + A_i^{\mathrm{T}} P_{i+1} A_i, \tag{15a}$$

$$H_i := Q_{xu,i} + A_i^{\mathrm{T}} P_{i+1} B_i, \tag{15b}$$

$$G_i := Q_{uu,i} + B_i^{\mathrm{T}} P_{i+1} B_i, \tag{15c}$$

$$\psi_{x,i} := h_{x,i} + A_i^{\mathrm{T}} P_{i+1} a_i + A_i^{\mathrm{T}} \Psi_{i+1}, \tag{15d}$$

$$\psi_{u,i} := h_{u,i} + B_i^{\mathrm{T}} P_{i+1} a_i + B_i^{\mathrm{T}} \Psi_{i+1}, \tag{15e}$$

and

$$\phi_{x,i} := A_i^{\mathrm{T}} \Phi_{i+1}, \quad \phi_{u,i} := B_i^{\mathrm{T}} \Phi_{i+1}. \tag{15f}$$

Second, $\Delta\lambda_{i+1}$ and Δx_{i+1} are eliminated from (10e) using (14a) and (10b). Subsequently, we can express Δu_i using Δx_i, Δt_k, and Δt_{k-1} as

$$\Delta u_i = K_i \Delta x_i + k_i + T_i(\Delta t_k - \Delta t_{k-1}) + W_i(-\Delta t_k), \tag{16a}$$

where

$$K_i := -G_i^{-1} H_i^{\mathrm{T}}, \tag{16b}$$

$$k_i := -G_i^{-1}(B_i^{\mathrm{T}} P_{i+1} \bar{x}_i - B_i^{\mathrm{T}} z_{i+1} + \bar{l}_{u,i}), \tag{16c}$$

and

$$T_i := -G_i^{-1}\psi_{u,i}, \quad W_i := -G_i^{-1}\phi_{u,i}. \tag{16d}$$

Third, $\Delta\lambda_{i+1}$, Δx_{i+1}, and Δu_i are eliminated from (10d) using (14a), (10b), and (16a), respectively. As a result, $\Delta\lambda_i$ using Δx_i, Δt_k, and Δt_{k-1} is expressed as

$$\Delta\lambda_i = P_i \Delta x_i - s_i + \Psi_i(\Delta t_k - \Delta t_{k-1}) + \Phi_i(-\Delta t_k), \tag{17a}$$

where

$$P_i := F_i - K_i^{\mathrm{T}} G_i K_i, \tag{17b}$$

$$s_i := -\left\{\bar{l}_{x,i} + A_i^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_{i+1}) + H_i k_i\right\}, \tag{17c}$$

and

$$\Psi_i := \psi_{x,i} + K_i \psi_{u,i}, \quad \Phi_i := \phi_{x,i} + K_i \phi_{u,i}. \tag{17d}$$

Moreover, by eliminating $\Delta\lambda_{i+1}$, Δx_{i+1}, and Δu_i from (14b) and (14c) using (14a), (10b), and (16a), we obtain:

$$\begin{aligned}&\sum_{j\in\mathcal{I}_k,\, j\leq i-1} \left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + a_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) \\ &\quad + \Psi_i^{\mathrm{T}}\Delta x_i + \xi_i(\Delta t_k - \Delta t_{k-1}) + \chi_i(-\Delta t_k) + \eta_i \\ &\quad - \Phi_i^{\mathrm{T}}\Delta x_i - \chi_i(\Delta t_k - \Delta t_{k-1}) - \rho_i(-\Delta t_k) - \iota_i = 0\end{aligned} \tag{18a}$$

and

$$\sum_{j\in\mathcal{I}_{k-1}}\left(h_{x,j}^{\mathrm{T}}\Delta x_j+h_{u,j}^{\mathrm{T}}\Delta u_j+a_j^{\mathrm{T}}\Delta\lambda_{j+1}+\bar{h}_j\right)-\sum_{j\in\mathcal{I}_k,\,j\le i-1}\left(h_{x,j}^{\mathrm{T}}\Delta x_j+h_{u,j}^{\mathrm{T}}\Delta u_j+a_j^{\mathrm{T}}\Delta\lambda_{j+1}+\bar{h}_j\right)+Q_{tt,k-1}(\Delta t_{k-1}-\Delta t_{k-2})+\bar{q}_{t,k-1}-\Psi_i^{\mathrm{T}}\Delta x_i-\xi_i(\Delta t_k-\Delta t_{k-1})-\chi_i(-\Delta t_k)-\eta_i=0, \tag{18b}$$

where

$$\xi_i:=\xi_{i+1}+a_i^{\mathrm{T}}(P_{i+1}a_i+2\Psi_{i+1})+\psi_{u,i}^{\mathrm{T}}T_i, \tag{19a}$$

$$\eta_i:=\eta_{i+1}+\bar{h}_i+a_i^{\mathrm{T}}(P_{i+1}\bar{x}_i-s_{i+1})+\Psi_{i+1}^{\mathrm{T}}\bar{x}_i+\psi_{u,i}^{\mathrm{T}}k_i, \tag{19b}$$

$$\chi_i:=\chi_{i+1}+\Phi_{i+1}^{\mathrm{T}}a_i+\psi_{u,i}^{\mathrm{T}}W_i, \tag{19c}$$

$$\rho_i:=\rho_{i+1}+\phi_{u,i}^{\mathrm{T}}W_i, \tag{19d}$$

and

$$\iota_i:=\iota_{i+1}+\Phi_{i+1}^{\mathrm{T}}\bar{x}_i+\phi_{u,i}^{\mathrm{T}}k_i. \tag{19e}$$

Therefore, we have Equations (17a), (18a), and (18b) for stage i having the same form as that of Equation (14) for stage $i+1$.

4.1.3 Phase transition stages

Here, we consider phase $k\in\{1,\dots,K\}$ and stage $i_k:=\min\mathcal{I}_k$. In this stage, the phase of interest changes from k to $k-1$ when $k>1$, and the backward recursion terminates when $k=1$. Equation (14) until i_k are expressed as

$$\Delta\lambda_{i_k}=P_{i_k}\Delta x_{i_k}-s_{i_k}+\Psi_{i_k}(\Delta t_k-\Delta t_{k-1})+\Phi_{i_k}(-\Delta t_k), \tag{20a}$$

$$\Psi_{i_k}^{\mathrm{T}}\Delta x_{i_k}+\xi_{i_k}(\Delta t_k-\Delta t_{k-1})+\chi_{i_k}(-\Delta t_k)+\eta_{i_k}-\Phi_{i_k}^{\mathrm{T}}\Delta x_{i_k}-\chi_{i_k}(\Delta t_k-\Delta t_{k-1})-\rho_{i_k}(-\Delta t_k)-\iota_{i_k}=0, \tag{20b}$$

and

$$\sum_{j\in\mathcal{I}_{k-1}}\left(h_{x,j}^{\mathrm{T}}\Delta x_j+h_{u,j}^{\mathrm{T}}\Delta u_j+a_j^{\mathrm{T}}\Delta\lambda_{j+1}+\bar{h}_j\right)+Q_{tt,k-1}(\Delta t_{k-1}-\Delta t_{k-2})+\bar{q}_{t,k-1}-\Psi_{i_k}^{\mathrm{T}}\Delta x_{i_k}-\xi_{i_k}(\Delta t_k-\Delta t_{k-1})-\chi_{i_k}(-\Delta t_k)-\eta_{i_k}=0. \tag{20c}$$

Note that $\Delta t_{k-1}=\Delta t_0=0$ when $k=1$. Therefore, from (20b), Δt_k is determined as

$$\Delta t_k=-\sigma_{i_k}^{-1}(\Psi_{i_k}-\Phi_{i_k})^{\mathrm{T}}\Delta x_{i_k}-\sigma_{i_k}^{-1}(\xi_{i_k}-\chi_{i_k})(-\Delta t_{k-1})-\sigma_{i_k}^{-1}(\eta_{i_k}-\iota_{i_k}). \tag{21}$$

Here, the following is defined:

$$\sigma_{i_k}:=\xi_{i_k}-2\chi_{i_k}+\rho_{i_k}. \tag{22}$$

The backward recursion is completed when $k=1$ because $i_1=\min\mathcal{I}_1=0$. When $k>1$, (21) is further substituted into (20a) and (20c), which produces:

$$\Delta\lambda_{i_K}=\tilde{P}_{i_K}\Delta x_{i_K}-\tilde{s}_{i_K}+\tilde{\Psi}_{i_K}(-\Delta t_{k-1}) \tag{23a}$$

and

$$\sum_{i\in\mathcal{I}_{K-1}}\left(h_{x,i}^{\mathrm{T}}\Delta x_i+h_{u,i}^{\mathrm{T}}\Delta u_i+a_i^{\mathrm{T}}\Delta\lambda_{i+1}+\bar{h}_i\right)+Q_{tt,k-1}(\Delta t_{k-1}-\Delta t_{k-2})+\bar{q}_{t,k-1}-\tilde{\Phi}_{i_k}^{\mathrm{T}}\Delta x_{i_k}-\tilde{\rho}_{i_k}(-\Delta t_{k-1})-\tilde{\iota}_{i_k}=0, \tag{23b}$$

where

$$\tilde{P}_{i_k}:=P_{i_k}-\sigma_{i_k}^{-1}(\Psi_{i_k}-\Phi_{i_k})(\Psi_{i_k}-\Phi_{i_k})^{\mathrm{T}}, \tag{23c}$$

$$\tilde{s}_{i_k}:=s_{i_k}+\sigma_{i_k}^{-1}(\eta_{i_k}-\iota_{i_k})(\Psi_{i_k}-\Phi_{i_k}), \tag{23d}$$

$$\tilde{\Phi}_{i_k}:=\Psi_{i_k}-\sigma_{i_k}^{-1}(\xi_{i_k}-\chi_{i_k})(\Psi_{i_k}-\Phi_{i_k}), \tag{23e}$$

$$\tilde{\rho}_{i_k}:=\xi_{i_k}-\sigma_{i_k}^{-1}(\xi_{i_k}-\chi_{i_k})^2, \tag{23f}$$

and

$$\tilde{\iota}_{i_k}:=\eta_{i_k}-\sigma_{i_k}^{-1}(\xi_{i_k}-\chi_{i_k})(\eta_{i_k}-\iota_{i_k}). \tag{23g}$$

Therefore, we obtain Equations (17a), (18a), and (18b) for stage i_k in the same form as Equation (14) for stage $i+1$ with $P_{i+1}=\tilde{P}_{i_k}$, $s_{i+1}=\tilde{s}_{i_k}$, $\Psi_{i+1}=0$, $\Phi_{i+1}=\tilde{\Psi}_{i_k}$, $\xi_{i+1}=Q_{tt,k-1}$, $\chi_{i+1}=0$, $\rho_{i+1}=\tilde{\rho}_{i_k}$, $\eta_{i+1}=\bar{q}_{t,k+1}$, and $\iota_{i+1}=\tilde{\iota}_{i_k}$.

4.2 Forward recursion

After the backward recursion up to the initial stage ($i=0$), all Newton steps are recursively computed from stage 0 to N forward in time using the results of the backward recursion. First, the Newton step of the initial state Δx_0 is computed from (10a). Second, the Newton step of the first switching time Δt_1 is computed from (21) with $\Delta t_0=0$. Third, $\Delta\lambda_i$, Δu_i, and Δx_{i+1} are computed forward in time in each phase k using (14a), (16a), and (10b), respectively. Fourth, after this procedure, Δt_{k+1} is computed using (21) until stage $i_{k+1}-1=\min\mathcal{I}_{k+1}-1$. The third and fourth steps are repeated from phase $k=1$ to $k=K+1$, and the forward recursion is completed by computing $\Delta\lambda_N$ using (13a).

The following lemma states the equivalence of the dynamic programming and the proposed Riccati recursion:

Lemma 4.1: *Consider a phase $k\in\mathcal{K}$ and stage $i\in\mathcal{I}_k$. Suppose that $G_j\succ O$ for all $j\ge i$, and $\sigma_{i_l}>0$ for all $l>k$, where G_j and σ_{i_l} are defined in* (15c) *and* (22)*, respectively. Then, the cost-to-go function of stage i of dynamic programming for the QP* (11) *is expressed as*

$$\frac{1}{2}\begin{bmatrix}-\Delta t_k\\ \Delta t_k-\Delta t_{k-1}\\ \Delta x_i\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\rho_i & \chi_i & \Phi_i^{\mathrm{T}}\\ \chi_i & \xi_i & \Psi_i^{\mathrm{T}}\\ \Phi_i & \Psi_i & P_i\end{bmatrix}\begin{bmatrix}-\Delta t_k\\ \Delta t_k-\Delta t_{k-1}\\ \Delta x_i\end{bmatrix}+\begin{bmatrix}\iota_i\\ \eta_i\\ -s_i\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}-\Delta t_k\\ \Delta t_k-\Delta t_{k-1}\\ \Delta x_i\end{bmatrix}, \tag{24}$$

where ξ_i, ρ_i, Φ_i, Ψ_i, P_i, ι_i, η_i, and s_i are defined by the Riccati recursion algorithm presented in subsection 4.1. Moreover, the subproblem of the dynamic programming to determine Δu_i is represented by

$$\begin{aligned}\min_{\Delta u_i} \quad & \frac{1}{2}\begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} Q_{tt,i} & h_{x,i}^{\mathrm{T}} & h_{u,i}^{\mathrm{T}} \\ h_{x,i} & Q_{xx,i} & Q_{xu,i} \\ h_{u,i} & Q_{ux,i} & Q_{uu,i} \end{bmatrix} \\ & \times \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} + \begin{bmatrix} \bar{h}_i \\ \bar{l}_{x,i} \\ \bar{l}_{u,i} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_i \\ \Delta u_i \end{bmatrix} \\ & + \frac{1}{2}\begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} \rho_{i+1} & \chi_{i+1} & \Phi_{i+1}^{\mathrm{T}} \\ \chi_{i+1} & \xi_{i+1} & \Psi_{i+1}^{\mathrm{T}} \\ \Phi_{i+1} & \Psi_{i+1} & P_{i+1} \end{bmatrix} \\ & \times \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix} + \begin{bmatrix} \iota_{i+1} \\ \eta_{i+1} \\ -s_{i+1} \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} -\Delta t_k \\ \Delta t_k - \Delta t_{k-1} \\ \Delta x_{i+1} \end{bmatrix} \\ \text{s.t.} \quad & A_i \Delta x_i + B_i \Delta u_i + a_i(\Delta t_k - \Delta t_{k-1}) - \Delta x_{i+1} + \bar{x}_i = 0, \end{aligned} \tag{25}$$

and determining Δt_k is expressed as

$$\min_{\Delta t_k} \quad (24). \tag{26}$$

Proof: The proof is achieved by induction. At the terminal stage ($i = N$), the cost-to-go function is represented by (24) with $\Psi_N = \Phi_N = 0$ and $\xi_N = \chi_N = \rho_N = \eta_N = \iota_N = 0$. Next, suppose that we have the cost-to-go function (24) of stage $i+1$. Then, it is clear that the subproblem of the dynamic programming is represented by (25). Δu_i can be uniquely determined from (25) as (16) because $G_i \succ O$. The cost-to-go function of stage i is represented by (24), where P_i, s_i, Ψ_i, Φ_i, ξ_i, χ_i, ρ_i, η_i, and ι_i are defined as (15), (16), (17), and (19). When $i = i_k := \min I_k$, Δt_k can be further uniquely determined by solving (26) under the assumption $\sigma_{i_k} > 0$ after determining Δu_{i_k} and obtaining the cost-to-go function of stage i_k (24). Then, the cost-to-go function of stage i_k is in the form of (24), where P_{i_k}, s_{i_k}, Φ_{i_k}, ρ_{i_k}, and ι_{i_k} are defined as (23), $\Psi_{i_k} = 0$, $\xi_{i_k} = Q_{tt,k-1}$, $\chi_{i_k} = 0$, and $\eta_{i_k} = \bar{q}_{t,k-1}$, respectively, which completes the proof. ■

Note that the equivalence cannot be shown without the positive definiteness of G_i and σ_{i_k}: if they are not positive definite, a unique solution does not exist and the cost-to-go function is not defined. The following theorem is obtained based on Lemma 4.1. Note that the discussion herein is restricted to the exact Hessian matrix to analyze the SOSC.

Theorem 4.2: *We suppose that the SOSC and LICQ hold at the current iterate and consider that the exact Hessian matrix is used. Then, G_i, as defined in* (15c)*, is positive definite for all $i \in \{0, \ldots, N-1\}$. Moreover, σ_{i_k}, as defined in* (22)*, satisfies $\sigma_{i_k} > 0$ for all $k \in \{0, \ldots, K\}$.*

Proof: The QP subproblem (11) with an exact Hessian matrix must have a unique global solution because the SOSC and LICQ hold (Nocedal & Wright, 2006). Therefore, the dynamic programming subproblems (25) and (26) must have unique solutions. Subsequently, the Hessian matrix with respect to Δu_i in (25) must be positive definite, and the quadratic term with respect to Δt_k in (26) must be positive. Then, Lemma 4.1 recursively shows that the Hessian matrix is G_i with respect to Δu_i. Additionally, the lemma also shows that the quadratic term with respect to Δt_k is σ_{i_k}, which completes the proof. ■

Theorem 4.2 indicates that G_i^{-1} can always be efficiently computed using Cholesky factorizations if the current iterate is sufficiently close to a local minimum. This fact also leads to the local (superlinear) convergence of the proposed method for NLP (4) under the SOSC and LICQ. The proof for this is omitted because it is trivial.

4.3 Reduced Hessian modification via Riccati recursion

The recused Hessian matrix can be indefinite when the solution is not sufficiently close to a local minimum, such that the SOSC does not hold. Subsequently, the KKT matrix is no longer invertible and the local convergence is not guaranteed. Efficient Cholesky factorization cannot be used to compute G_i^{-1}. Therefore, an algorithmic modification of the Riccati recursion is proposed to make the algorithm numerically robust and efficient for such situations, which can be considered a modification on the reduced Hessian matrix. To consider practical situations, Hessian approximations are allowed in the following, while Theorem 4.2 analyzes the exact Hessian matrix. First, we introduce the following practical assumption:

Assumption 4.1: $\begin{bmatrix} Q_{xx,i} & Q_{xu,i} \\ Q_{xu,i}^{\mathrm{T}} & Q_{uu,i} \end{bmatrix} \succeq O$ *and* $Q_{uu,i} \succ O$ *for all* $i \in \{0, \ldots, N-1\}$, $Q_{tt,k} \geq 0$ *for all* $k \in \{0, \ldots, K\}$, *and* $Q_{xx,N} \succeq O$.

This assumption is easily satisfied with the Gauss-Newton Hessian approximation or, more generally, with sequential convex programming (Messerer et al., 2021) for $\nabla_{xx} H_k(x_i, u_i, \lambda_{i+1})$, $\nabla_{xu} H_k(x_i, u_i, \lambda_{i+1})$, and $\nabla_{uu} H_k(x_i, u_i, \lambda_{i+1})$. Under Assumption 4.1, a unique solution of the dynamic programming subproblem (25) exists (that is, $G_i \succ O$) and $P_i \succeq O$ if $P_{i+1} \succeq O$, which is the same discussion as the dynamic programming for standard linear quadratic OCPs (Bertsekas, 2005). The solution to (26) also exists if $\sigma_{i_k} > 0$. Based on these observations, a modification of the Riccati recursion algorithm is proposed, whereby $\tilde{P}_{i_k}$ is updated at each phase-transition stage $i_k = \min \mathcal{I}_k$. This is expressed as

$$\tilde{P}_{i_k} = P_{i_k} \tag{27}$$

instead of (23c). Then, $\tilde{P}_{i_k} \succeq O$, as long as $P_{i_k} \succeq O$. Note that a different method of (27) can be used to make $\tilde{P}_{i_k} \succeq O$, which eliminates the negative curvature from P_{i_k} through eigenvalue decomposition (Quirynen et al., 2014). However, this requires considerably more computational time than (27). Nevertheless, the computationally cheap modification (27) works surprisingly

well in practise. We also modify σ_{i_k} instead of (22), which is expressed as

$$\tilde{\sigma}_{i_k} = \begin{cases} \sigma_{i_k} & \sigma_{i_k} > \sigma_{\min} \\ |\sigma_{i_k}| + \bar{\sigma} & \sigma_{i_k} \leq \sigma_{\min} \end{cases}, \tag{28}$$

where $\sigma_{\min} \geq 0$ and $\bar{\sigma} > 0$. A practical rule to choose $\sigma_{\min}$ and $\bar{\sigma}$ is as follows. We empirically observed that numerical ill-conditioning in the Newton-type method produced a large Δt_s, that is, a too small σ_{i_k} in (21). From this observation, a desired maximum absolute value of Δt_s was chosen as $\Delta t_{s,\max} > 0$. Subsequently, $\sigma_{\min}$ and $\bar{\sigma}$ were chosen such that the absolute value of $\sigma_{i_k}^{-1}(\eta_{i_k} - \iota_{i_k})$ did not exceed $\Delta t_{s,\max}$. This is expressed as

$$\sigma_{\min} = \bar{\sigma} = \Delta t_{s,\max}^{-1} |\eta_{i_k} - \iota_{i_k}| \tag{29}$$

for each $k \in \mathcal{K}$. For example, it was discovered that choosing $\Delta t_{s,\max}$ from a range of 0.1 to 1.0 worked well in practise.

The following Theorem claims that the proposed algorithmic modification makes the reduced Hessian matrix of the KKT system (10) positive definite and the KKT matrix invertible:

Theorem 4.3: *Suppose that Assumption 4.1 holds. Then, the reduced Hessian matrix of the KKT system* (10a) *with the modifications* (27) *and* (28) *is positive definite. If, in addition, the LICQ holds, then the KKT matrix is invertible.*

Proof: Under Assumption 4.1 and with the Hessian modifications (27) and (28), $P_i \succeq O$ for all $i = 0, \ldots, N$, $G_i \succ O$ for all $i = 0, \ldots, N-1$, and $\sigma_{i_k} > 0$ for all $k = 1, \ldots, K$ hold. Therefore, the QP subproblems of the dynamic programming (25) and (26) always have unique global solutions. Theorem 16.2 of Nocedal and Wright (2006) completes the first claim. The second claim then directly follows from Theorem 16.1 of Nocedal and Wright (2006). ■

Remark 4.1: The KKT matrix is always invertible under Assumption 4.1 and the proposed reduced Hessian modifications. Therefore, the local convergence of the Newton-type method is ensured if the resultant reduced Hessian is sufficiently close to the exact one (Messerer et al., 2021; Nocedal & Wright, 2006) and the LICQ holds.

4.4 Summary of algorithm

The single Newton iteration using the proposed Riccati recursion algorithm is summarised in Algorithm 2. After computing the KKT system (10) (line 1), the proposed method recursively eliminates the Newton steps from the KKT system (10) in the backward recursion (lines 3–14) and recursively computes the Newton steps in the forward recursion (lines 16–27). Finally, the Newton steps of the slack variables and Lagrange multipliers are computed from the other Newton steps (line 28) according to the primal–dual interior point method (Nocedal & Wright, 2006; Wächter & Biegler, 2006). Calculations in the proposed algorithm, such as matrix inversions or matrix–matrix multiplications, do not grow with respect to the length of the horizon N. Therefore, the computational time of the proposed method is $O(N)$ as the standard Riccati recursion algorithm (Rao et al., 1998). After computing the Newton steps, we can employ typical line-search methods, such as the merit function-based methods or filter-based methods (Nocedal & Wright, 2006), for the smooth constrained NLP (6).

Algorithm 2 Computation of Newton step via proposed Riccati recursion

Require: Initial state $x(t_0)$ and the current iterate $x_0, \ldots, x_N, x_s, u_0, \ldots, u_{N-1}, u_s, \lambda_0, \ldots, \lambda_N, \lambda_s$, and t_s.

Ensure: Newton steps $\Delta x_0, \ldots, \Delta x_N, \Delta x_s, \Delta u_0, \ldots, \Delta u_{N-1}, \Delta u_s, \Delta\lambda_0, \ldots, \Delta\lambda_N, \Delta\lambda_s$, and Δt_s.

1: Compute the KKT system (10).
2: `// Backward recursion`
3: Compute P_N and z_N from (13).
4: **for** $k = K+1, \ldots, 1$ **do**
5: **for** $i = \max \mathcal{I}_k, \ldots, \min \mathcal{I}_k$ **do**
6: Compute K_i, k_i, T_i, and W_i from (16) .
7: Compute P_i, z_i, Ψ_i, and Φ_i from (17).
8: Compute ξ_i, ξ_i, ρ_i, η_i, and ι_i, from (19).
9: **end for**
10: **if** $k < K+1$ **then**
11: Compute $\tilde{P}_{i_k}$, $\tilde{s}_{i_k}$, $\tilde{\Phi}_{i_k}$, $\tilde{\rho}_{i_k}$, and $\tilde{\iota}_{i_k}$ from (23) optionally with modifications (27) and (28).
12: **end if**
13: Set Ψ_{i_k}, ξ_{i_k}, χ_{i_k}, and ι_{i_k} to zero.
14: **end for**
15: `// Forward recursion`
16: Compute Δx_0 from (10a).
17: Compute Δt_1 from (21).
18: **for** $k = 1, \ldots, K+1$ **do**
19: **for** $i = \min \mathcal{I}_k, \ldots, \max \mathcal{I}_k$ **do**
20: Compute Δu_i and $\Delta\lambda_i$ from (16) and (17a), respectively.
21: Compute Δx_{i+1} from (10b)
22: **end for**
23: **if** $k < K+1$ **then**
24: Compute Δt_{k+1} from (21)
25: **end if**
26: **end for**
27: Compute $\Delta\lambda_N$ from (13).
28: Compute Δz_i, Δv_i, Δw_i, and $\Delta \upsilon_i$ for $i = 0, \ldots, N-1$ from the other Newton steps.

5. State jumps and switching conditions

In this section, the proposed NLP formulations and Riccati recursion algorithm are further extended to switched systems involving state jumps and switching conditions, which have not yet been considered. Some classes of switched systems involve state jumps at the same time as the switch, which is expressed as

$$x(t_k) = f_j(x(t_k-)), \quad f_j : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}, \tag{30}$$

where t_k- is the instant immediately before t_k. We also consider that there is a stage cost on the state immediately before the state jump. That is, it is assumed that $l_j(x(t_k-))$ is added to the cost function (3a). Moreover, such switches are often state-dependent, that is, the switch occurs if the state satisfies some

condition (hereafter called the switching condition), which is expressed as

$$e(x(t_k-)) = 0, \quad e : \mathbb{R}^{n_x} \to \mathbb{R}^{n_e}. \tag{31}$$

For example, a legged robot can be modeled as a system with switches involving state jumps and switching conditions. When the distance between the robot's foot and the ground becomes zero (the switching condition), the generalised velocity instantly changes due to the impact (the state jump) (Farshidian et al., 2017; Li & Wensing, 2020).

If there is a state jump at the kth switch, a new grid point i_k- is introduced corresponding to time t_k-, which is added to the KKT conditions and expressed as

$$x_{i_k} = f_j(x_{i_k-}) \tag{32a}$$

and

$$\nabla_x l_j(x_{i_k-}) + \nabla_x f_j^{\mathrm{T}}(x_{i_k-})\lambda_{i_k} - \lambda_{i_k-} = 0. \tag{32b}$$

The KKT systems regarding the state jump are therefore expressed as

$$\Delta x_{i_k} = A_{i_k-}\Delta x_{i_k-} + \bar{x}_{i_k-} \tag{33a}$$

and

$$\Delta\lambda_{i_k-} = Q_{xx,i_k-}\Delta x_{i_k-} + A_{i_k-}^{\mathrm{T}}\Delta\lambda_{i_k} + \bar{l}_{x,i_k-}, \tag{33b}$$

where $A_{i_k-} := \nabla_x f_j(x_{i_k-})$ and Q_{xx,i_k-} is the Hessian of the Lagrangian with respect to x_{i_k-}. $\bar{x}_{i_k}$ and $\bar{l}_{x,i_k-}$ are the residuals in (32a) and (32b), respectively. When (33a) and (33b) are considered in the backward recursion, we have the equations until stage $i_k = \min \mathcal{I}_k$. That is, the equations are in the form of (23a), (23b), and (10f), with the phase index k replaced with $k-1$. By eliminating $\Delta\lambda_{i_k}$ and Δx_{i_k} from these equations using (33a) and (33b), we then obtain

$$\Delta\lambda_{i_k-} = P_{i_k-}\Delta x_{i_k-} + \Phi_{i_k-}(-\Delta t_{k-1}) - s_{i_k-}, \tag{34a}$$

$$\begin{aligned}
&\sum_{i\in\mathcal{I}_{k-1}} \left(h_{x,i}^{\mathrm{T}}\Delta x_i + h_{u,i}^{\mathrm{T}}\Delta u_i + a_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{h}_i\right) \\
&\quad + Q_{tt,k-1}(\Delta t_{k-1} - \Delta t_{k-2}) + \bar{q}_{t,k-1} \\
&\quad - \Phi_{i_k-}^{\mathrm{T}}\Delta x_{i_k-} - \rho_{i_k-}(\Delta t_{k-1}) - \eta_{i_k-} = 0,
\end{aligned} \tag{34b}$$

and

$$\begin{aligned}
&\sum_{i\in\mathcal{I}_{k-2}} \left(h_{x,i}^{\mathrm{T}}\Delta x_i + h_{u,i}^{\mathrm{T}}\Delta u_i + a_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{h}_i\right) \\
&\quad - \sum_{i\in\mathcal{I}_{k-1}} \left(h_{x,i}^{\mathrm{T}}\Delta x_i + h_{u,i}^{\mathrm{T}}\Delta u_i + a_i^{\mathrm{T}}\Delta\lambda_{i+1} + \bar{h}_i\right) \\
&\quad + Q_{tt,k-2}(\Delta t_{k-2} - \Delta t_{k-3}) - Q_{tt,k-1}(\Delta t_{k-1} - \Delta t_{k-2}) \\
&\quad + \bar{q}_{t,k-2} - \bar{q}_{t,k-1} = 0,
\end{aligned} \tag{34c}$$

where

$$P_{i_k-} := Q_{xx,i_k-} + A_{i_k-}^{\mathrm{T}}\tilde{P}_{i_k}A_{i_k-}, \tag{34d}$$

$$\Phi_{i_k-} := A_{i_k-}^{\mathrm{T}}\tilde{\Phi}_{i_k}, \tag{34e}$$

$$s_{i_k-} := A_{i_k-}^{\mathrm{T}}(\tilde{s}_{i_k} - \tilde{P}_{i_k}\bar{x}_{i_k-}) - \bar{l}_{x,i_k-}, \tag{34f}$$

and

$$\rho_{i_k-} = \tilde{\rho}_{i_k}, \quad \eta_{i_k-} = \tilde{\eta}_{i_k} + \tilde{\Phi}_{i_k}^{\mathrm{T}}\bar{x}_{i_k-}. \tag{34g}$$

Therefore, the same equation as (17) is achieved at stage i_k-, and we can proceed to stage $i_k - 1$. In the forward recursion, Δx_{i_k-} is computed from Δx_{i_k-1} and Δu_{i_k-1} using (10b) at stage $i_k - 1$. Then, Δx_{i_k} and $\Delta\lambda_{i_k-}$ are computed from Δx_{i_k-} according to (33a) and (17a) for $i = i_k-$ with $\Psi_{i_k} = 0$.

5.1 Switching conditions

The switching conditions are typically presented as pure-state equality constraints, which are difficult to treat efficiently with the Riccati recursion algorithm (specifically with the $O(N)$ computational burden) (Sideris & Rodriguez, 2011). In this section, we assume that the state is partitioned into the generalised coordinate and velocity as $x = [\, q^{\mathrm{T}}\ v^{\mathrm{T}}\,]^{\mathrm{T}}$, $q, v \in \mathbb{R}^n$ and the state equation of each subsystem is expressed as

$$f(x,u) := \begin{bmatrix} f^{(q)}(x) \\ f^{(v)}(x,u) \end{bmatrix}, \tag{35}$$

where $f^{(q)} : \mathbb{R}^{n_x} \to \mathbb{R}^n$ and $f^{(v)} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^n$. Moreover, we assume that the pure-state equality constraints representing the switching conditions have a relative degree of two, that is, the Jacobian of (31) is expressed as

$$\nabla_x e(x) = \begin{bmatrix} \nabla_q e(q) & O \end{bmatrix}, \quad \nabla_q e(q) \in \mathbb{R}^{n_e \times n}. \tag{36}$$

These assumptions mainly represent the position-level constraints on mechanical systems, such as robotic systems having rigid contacts. To use the Newton-type method with $O(N)$ complexity under these assumptions, we did not consider the pure-state equality constraint

$$e(x_{i_k}-) = 0 \tag{37}$$

directly as was considered by Sideris and Rodriguez (2011). Instead, the constraint was transformed into a mixed state-input constraint at the two-stage grid point before the switch (Katayama & Ohtsuka, 2021a), which is expressed as

$$e_i := e\left(x_i + f(x_i,u_i)\Delta\tau_k + g(x_i + f(x_i,u_i)\Delta\tau_k)\Delta\tau_k\right) = 0, \tag{38}$$

where i is two-stage before the grid i_k-, that is, i satisfies $i + 2 = i_k-$. The Lagrange multiplier is introduced with respect to (38), $\zeta_i \in \mathbb{R}^{n_e}$. The KKT conditions regarding this stage are then represented by (4c), (38), $\tilde{r}_{x,i} := r_{x,i} + C_i^{\mathrm{T}}\zeta_i = 0$, and $\tilde{r}_{u,i} := r_{u,i} + D_i^{\mathrm{T}}\zeta_i = 0$, where $C_i := \nabla_x e(\cdot)$ and $D_i := \nabla_u e(\cdot)$. Furthermore, the KKT conditions of (9f) related to t_{k-1} are replaced with:

$$\begin{aligned}
&E_i^{\mathrm{T}}\zeta_i + \frac{1}{N_{k-1}}\sum_{j\in\mathcal{I}_{k-1}} H_{k-1}(x_j,u_j,\lambda_{j+1}) \\
&\quad - \frac{1}{N_k}\sum_{i\in\mathcal{I}_k} H_k(x_j,u_j,\lambda_{j+1}) - \upsilon_{k-1} + \upsilon_k = 0
\end{aligned}$$

and

$$\frac{1}{N_{k-2}}\sum_{j\in\mathcal{I}_{k-2}} H_{k-2}(x_j,u_j,\lambda_{j+1}) - E_i^{\mathrm{T}}\zeta_i - \frac{1}{N_{k-1}}\sum_{i\in\mathcal{I}_{k-1}} H_{k-1}(x_j,u_j,\lambda_{j+1}) - \upsilon_{k-2} + \upsilon_{k-1} = 0,$$

where $E_i := \frac{1}{N_{k-1}}\nabla_{t_k} e(\cdot)$. The KKT systems regarding this stage are then expressed as (10b),

$$C_i\Delta x_i + D_i\Delta u_i + E_i(\Delta t_k - \Delta t_{k-1}) + \bar{e}_i = 0, \tag{39a}$$

$$Q_{xx,i}\Delta x_i + Q_{xu,i}\Delta u_i + A_i^{\mathrm{T}}\Delta\lambda_{i+1} + C_i^{\mathrm{T}}\Delta\zeta_i - \Delta\lambda_i + h_{x,i}(\Delta t_k - \Delta t_{k-1}) + \tilde{l}_{x,i} = 0, \tag{39b}$$

and

$$Q_{xu,i}^{\mathrm{T}}\Delta x_i + Q_{uu,i}\Delta u_i + B_i^{\mathrm{T}}\Delta\lambda_{i+1} + D_i^{\mathrm{T}}\Delta\zeta_i + h_{u,i}(\Delta t_k - \Delta t_{k-1}) + \tilde{l}_{u,i} = 0, \tag{39c}$$

where

$$\tilde{l}_{x,i} := \tilde{r}_{x,i} + \nabla_x g^{\mathrm{T}}(x_i,u_i)\mathrm{diag}(z_i)^{-1}(\mathrm{diag}(\nu_i)r_{g,i} - r_{z,i})$$

and

$$\tilde{l}_{u,i} := \tilde{r}_{u,i} + \nabla_u g^{\mathrm{T}}(x_i,u_i)\mathrm{diag}(z_i)^{-1}(\mathrm{diag}(\nu_i)r_{g,i} - r_{z,i}).$$

In addition, $\bar{e}_i$ is residual in (38).

In the backward recursion at stage i, we have (14a),

$$\sum_{j\in\mathcal{I}_{k-1},\, j\le i}\left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + f_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) + E_i^{\mathrm{T}}(\Delta\zeta_i + \zeta_i) + \Psi_{i+1}^{\mathrm{T}}\Delta x_{i+1} + \xi_{i+1}(\Delta t_{k-1} - \Delta t_{k-2}) + \chi_{i+1}(-\Delta t_{k-1}) + \eta_{i+1} - \Phi_{i+1}^{\mathrm{T}}\Delta x_{i+1} - \chi_{i+1}(\Delta t_{k-1} - \Delta t_{k-2}) - \rho_{i+1}(-\Delta t_{k-1}) - \iota_{i+1} = 0, \tag{40a}$$

and

$$\sum_{j\in\mathcal{I}_{k-2}}\left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + f_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) - \sum_{j\in\mathcal{I}_{k-1},\, j\le i}\left(h_{x,j}^{\mathrm{T}}\Delta x_j + h_{u,j}^{\mathrm{T}}\Delta u_j + f_j^{\mathrm{T}}\Delta\lambda_{j+1} + \bar{h}_j\right) + Q_{tt,k-2}(\Delta t_{k-2} - \Delta t_{k-3}) + \bar{q}_{t,k-2} - \Psi_{i+1}^{\mathrm{T}}\Delta x_{i+1} - \xi_{i+1}(\Delta t_{k-1} - \Delta t_{k-2}) - \chi_{i+1}(-\Delta t_{k-1}) - \eta_{i+1} + E_i^{\mathrm{T}}(\Delta\zeta_i + \zeta_i) = 0. \tag{40b}$$

Δu_i and $\Delta\zeta_i$ are then eliminated, resulting in:

$$\begin{bmatrix}\Delta u_i\\ \Delta\zeta_i\end{bmatrix} = \begin{bmatrix}K_i^{\mathrm{T}}\\ M_i\end{bmatrix}\Delta x_i + \begin{bmatrix}T_i\\ L_i\end{bmatrix}(\Delta t_{k-1} - \Delta t_{k-2}) + \begin{bmatrix}W_i\\ N_i\end{bmatrix}(-\Delta t_{k-1}) + \begin{bmatrix}k_i\\ m_i\end{bmatrix}, \tag{41a}$$

where

$$\begin{bmatrix}K_i\\ M_i\end{bmatrix} := -\begin{bmatrix}G_i & D_i^{\mathrm{T}}\\ D_i & \end{bmatrix}^{-1}\begin{bmatrix}H_i^{\mathrm{T}}\\ C_i\end{bmatrix}, \tag{41b}$$

$$\begin{bmatrix}T_i\\ L_i\end{bmatrix} := -\begin{bmatrix}G_i & D_i^{\mathrm{T}}\\ D_i & \end{bmatrix}^{-1}\begin{bmatrix}\psi_{u,i}\\ E_i\end{bmatrix}, \tag{41c}$$

$$\begin{bmatrix}W_i\\ N_i\end{bmatrix} := -\begin{bmatrix}G_i & D_i^{\mathrm{T}}\\ D_i & \end{bmatrix}^{-1}\begin{bmatrix}\phi_{u,i}\\ 0\end{bmatrix}, \tag{41d}$$

and

$$\begin{bmatrix}k_i\\ m_i\end{bmatrix} := -\begin{bmatrix}G_i & D_i^{\mathrm{T}}\\ D_i & \end{bmatrix}^{-1}\begin{bmatrix}B_i{}^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_i) + \tilde{l}_{u,i}\\ \bar{e}_i\end{bmatrix}. \tag{41e}$$

Therefore, we obtain the form equations of (17a), (18a), and (18b) for stage i with the following:

$$P_i = F_i - \begin{bmatrix}K_i\\ M_i\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}G_i & D_i^{\mathrm{T}}\\ D_i & \end{bmatrix}\begin{bmatrix}K_i\\ M_i\end{bmatrix}, \tag{42a}$$

$$s_i = -\left\{\tilde{l}_{x,i} + A_i^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_{i+1}) + H_i k_i + C_i^{\mathrm{T}} m_i\right\}, \tag{42b}$$

and

$$\Psi_i = \psi_{x,i} + \begin{bmatrix}K_i\\ M_i\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\psi_{u,i}\\ E_i\end{bmatrix}, \quad \Phi_i = \phi_{x,i} + \begin{bmatrix}K_i\\ M_i\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\phi_{u,i}\\ 0\end{bmatrix}. \tag{42c}$$

$$\xi_i = \xi_{i+1} + a_i^{\mathrm{T}}(P_{i+1}a_i + 2\Psi_{i+1}) + \psi_{u,i}^{\mathrm{T}}T_i + E_i^{\mathrm{T}}L_i, \tag{43a}$$

$$\eta_i = \eta_{i+1} + \bar{h}_i + E_i^{\mathrm{T}}\zeta_i + a_i^{\mathrm{T}}(P_{i+1}\bar{x}_i - s_{i+1}) + \Psi_{i+1}^{\mathrm{T}}\bar{x}_i + \psi_{u,i}^{\mathrm{T}}k_i + E_i^{\mathrm{T}}m_i, \tag{43b}$$

$$\chi_i = \chi_{i+1} + \Phi_{i+1}^{\mathrm{T}}a_i + \psi_{u,i}^{\mathrm{T}}W_i + E_i^{\mathrm{T}}N_i, \tag{43c}$$

$$\rho_i = \rho_{i+1} + \phi_{u,i}^{\mathrm{T}}W_i, \tag{43d}$$

and

$$\iota_i = \iota_{i+1} + \Phi_{i+1}^{\mathrm{T}}\bar{x}_i + \phi_{u,i}^{\mathrm{T}}k_i. \tag{43e}$$

At the forward recursion, Δu_i and $\Delta\zeta_i$ are computed using (41a) instead of only computing Δu_i as (16a).

5.2 Summary of algorithm

We herein summarise the proposed algorithm for the OCPs with state jumps and switching conditions as an extension of Algorithm 2. The overall algorithm is the same as Algorithm 2 except for the computations around $i = i_{k-}$ and $i = i_{k-} - 2$ for $k = 1,\dots,K+1$ The computation of the KKT system (line 1 of Algorithm 2) now includes the computation regarding state jumps such as (33a) and (33b) when $i = i_{k-}$ and these regarding switching conditions such as (39a)–(39c) when $i = i_{k-} - 2$ for $k = 1,\dots,K+1$. In the backward recursion (lines 3–14 of Algorithm 2), we compute the terms such as P_i and s_i using (34d)–(34g) when $i = i_{k-}$ and (42a)–(43e) when $i = i_{k-} - 2$ for $k = 1,\dots,K+1$. In the forward recursion (lines 16–27 of Algorithm 2), we additionally utilise (33a) and (33b) to compute $\Delta x_{i_{k-}}$ and $\Delta\lambda_{i_{k-}}$ when $i = i_{k-}$ and (41a) to compute Δu_i and $\Delta\zeta_i$ when $i = i_{k-} - 2$ for $k = 1,\dots,K+1$.

6. Numerical experiments

6.1 Comparison with off-the-shelf solvers

6.1.1 Problem settings

The effectiveness of the proposed method was demonstrated through two numerical experiments. The first experiment was a comparison with off-the-shelf NLP solvers on the switched system, consisting of three nonlinear subsystems treated in Xu and Antsaklis (2002, 2004). The dynamics of the subsystems are expressed as

$$f_1(x,u) = \begin{bmatrix} x_1 + u_1 \sin(x_1) \\ -x_2 - u_1 \cos(x_2) \end{bmatrix},$$

$$f_2(x,u) = \begin{bmatrix} x_2 + u_1 \sin(x_2) \\ -x_1 - u_1 \cos(x_1) \end{bmatrix},$$

and

$$f_3(x,u) = \begin{bmatrix} -x_1 - u_1 \sin(x_1) \\ x_2 + u_1 \cos(x_2) \end{bmatrix}.$$

The stage cost is expressed as

$$l_k(x,u) = \frac{1}{2}\|x - x_{\mathrm{ref}}\|^2 + \|u\|^2, \quad k \in \{1,2,3\}.$$

In addition, the terminal cost is expressed as

$$V_f(x) = \frac{1}{2}\|x - x_{\mathrm{ref}}\|^2,$$

where $x_{\mathrm{ref}} = [1,\ -1]^{\mathrm{T}}$. The initial and terminal times of the horizon are denoted by $t_0 = 0$ and $t_3 = 3$, respectively. The initial state is denoted by $x(t_0) = [2,\ 3]^{\mathrm{T}}$.

Benchmarks were set for the solution times of the NLP (4) example against the Ipopt (Wächter & Biegler, 2006) and a SQP methods with qpOASES (Ferreau et al., 2014), both of which were used through the CasADi (Andersson et al., 2019) interface. The mesh-refinement was not considered in this comparison to focus on a pure comparison of the abilities to solve the NLP problems. That is, only the single NLP step of Algorithm 1 was considered. Ipopt was used with the default settings of CasADi. For example, MUMPS (Amestoy et al., 2001) was used to solve the linear systems. The built-in SQP solver of CasADi was used with the exact Hessian matrix and qpOASES as the backend QP solver. The Hessian regularization was enabled within the qpOASES options and the Hessian type was set to 'indefinite'. The proposed method was written in C++, used Eigen (Guennebaud & Jacob, 2010) for the linear algebra, and used an exact Hessian matrix and reduced Hessian modifications (27) and (28) throughout the experiments. $\sigma_{\min}$ and $\bar{\sigma}$ were chosen using (29) with $\Delta t_{k,\max} = 0.5$. The proposed method only used the fraction-to-boundary rule (Nocedal & Wright, 2006; Wächter & Biegler, 2006) for the step size selection and monotonically decreased the barrier parameter ϵ when the l_2 norm of the KKT residual (in the perturbed KKT conditions (9)) was smaller than 0.1. The solution times were compared for these algorithms with the total number of grids set as $N = 10, 50, 100$, and 500. N was divided into N_1, N_2, and N_3, so that each phase had an almost equal number of discretization grids. The values of $[N_1, N_2, N_3]$ were set to $[4, 3, 3]$, $[17, 17, 16]$, $[34, 33, 33]$, and $[167, 167, 166]$ for $N = 10$, 50, 100, and 500, respectively. The minimum dwell-times were set as $\underline{\Delta}_1 = \underline{\Delta}_2 = \underline{\Delta}_3 = 0.01$ in the constraints (4e) to avoid ill-conditioned problems. The initial guess of the solution was set as $x_0 = x_1 = \cdots = x_N = x(t_0)$, $u_0 = u_1 = \cdots u_{N-1} = 0$, $t_1 = 1.0$, and $t_2 = 2.0$ for all solvers. It was assumed that the proposed method converged when the solution satisfied the default convergence criteria of Ipopt (max norm of the residual in the unperturbed KKT conditions (Nocedal & Wright, 2006; Wächter & Biegler, 2006)). All solvers were run 100 times for each N on a laptop with a quad-core CPU Intel Core i7-10510U @1.80 GHz, and the average computational time until convergence was measured.

Table 1. Average computational time (ms) of each solver for different total number of grids.

N	Proposed	Ipopt	SQP with qpOASES
10	0.08	4.3	failed
50	0.27	24.6	failed
100	0.47	45.1	failed
500	1.93	127	failed

6.1.2 Results

The average computational times of each solver for the different total number of grids N are listed in Table 1. As shown in the table, the proposed method converged the fastest of all the cases, and was up to two orders of magnitude faster than Ipopt. The proposed method was extremely fast because the dimension of the control input of each subsystem was 1 and it did not involve any explicit matrix inversions. The SQP method with qpOASES failed to converge for all cases, even the regularization and indefinite-Hessian mode were enabled. Note that another QP solver OSQP (Stellato et al., 2020) was also tested, but it failed to treat the indefinite Hessian matrix.

The total computational time of Algorithm 1 until convergence was also measured, including the mesh-refinement steps. The thresholds for the discretization step sizes were $\Delta\tau_{\max} = 0.35, 0.065, 0.035$, and 0.0065 for cases $N = 10$, 50, 100, and 500, respectively. The total computational times of Algorithm 1 for $N = 10$, 50, 100, and 500, were 0.13, 0.57, 0.92, and 4.3 ms, respectively. The mesh-refinements were only conducted a few times in each case, and a similar solution was achieved compared to that of the continuous-time counterpart reported in Xu and Antsaklis (2004). As expected, the solution accuracy improved as N increased.

6.2 Whole-body optimal control of quadrupedal gaits

6.2.1 Problem settings

Numerical experiments on the whole-body optimal control of a quadrupedal robot ANYmal (Hutter et al., 2017) were conducted to demonstrate the efficiency of the proposed method in practical and complicated examples. A trotting motion with a step length of 0.15 m and aggressive jumping motion with a jump length of 0.8 m also investigated. The contact patterns of the trotting and jumping motions are shown in Figure 2. The state equation of the robot switched based on the combination of the support feet, that is, it switched when the feet lifted off from the ground or touched down onto the ground. Therefore, the trotting motion had four switches and the jump

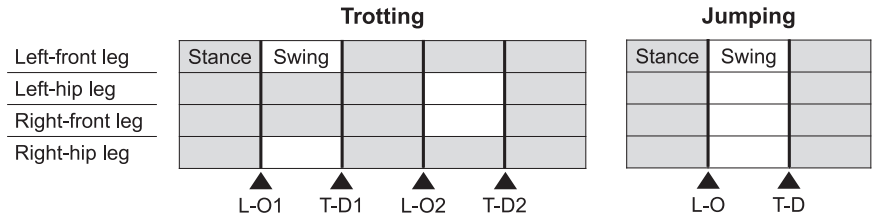

Figure 2. Contact patterns of trotting and jumping motions. Gray and white cells indicate the intervals where the leg is standing and swinging, respectively. Thick lines illustrate the switches of the active subsystems. Black triangles indicate the types of switches (lift-off: L-O, touch-down: T-D).

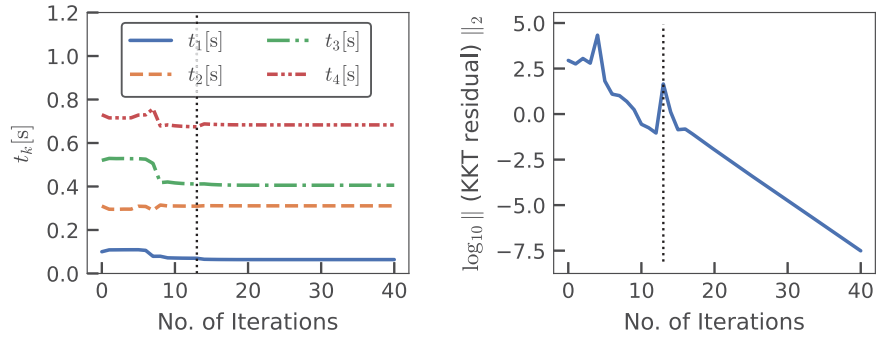

Figure 3. Convergence of the proposed method for the trotting motion of a quadrupedal robot: (a) switching instants and (b) l_2 norm of the residual in the perturbed KKT conditions over the iterations. Vertical dotted lines indicate that the mesh refinement was performed.

motion had two switches over the horizon, as shown in Figure 2. The lift-off did not include the state jump or the switching condition, but the touch-down included both (Farshidian et al., 2017; Li & Wensing, 2020; Schultz & Mombaur, 2010). The switching condition was treated using the proposed method in Section 5.1: the switching conditions were the position constraints on the foot whose Jacobian took the form of (36), and the state equation of the robot under an acceleration-level rigid-contact constraint took the form of (35). The details of the above formulations can be found in Li and Wensing (2020); Schultz and Mombaur (2010). Note also that the acceleration-level contact-consistent dynamics of the robot were lifted to improve the convergence speed without changing the structure of the KKT system (10) (Katayama & Ohtsuka, 2021b). To consider a practical situation, the limitations that were imposed on the joint angles, velocities, and torques were considered to be the inequality constraints. The polyhedral-approximated friction cone constraint was also considered for each contact force expressed in the world frame $[f_x\ f_y\ f_z]$ as:

$$\begin{bmatrix} f_x + \frac{\mu}{\sqrt{2}} f_z \\ -f_x + \frac{\mu}{\sqrt{2}} f_z \\ f_y + \frac{\mu}{\sqrt{2}} f_z \\ -f_y + \frac{\mu}{\sqrt{2}} f_z \\ f_z \end{bmatrix} \geq 0, \tag{44}$$

where $\mu > 0$ is the friction coefficient, which was set as $\mu = 0.7$. Note that the friction cone constraint (44) also switched depending on the active subsystems (that is, depending on the combination of the support feet). The initial and terminal times of the horizon were set as $t_0 = 0$ and $t_5 = 1.0$ for the trotting problem and $t_0 = 0$ and $t_2 = 1.7$ for the jumping problem, respectively.

To design the stage cost of the trotting motion, the state $x \in \mathbb{R}^{36}$ was partitioned into the generalised coordinate and velocity $[q^{\mathrm{T}}\ v^{\mathrm{T}}]^{\mathrm{T}}$, $q, v \in \mathbb{R}^{18}$. Additionally, the position (i.e. forward kinematics) of the foot $i \in$ Left-front, Left-hip, Right-front, Right-hip was considered in the world frame $p_i(q) \in \mathbb{R}^3$. The stage cost was then designed for the trotting motion, which is expressed as

$$\begin{aligned} &\frac{1}{2}(q - q_{\mathrm{ref}})^{\mathrm{T}} W_q (q - q_{\mathrm{ref}}) \\ &\quad + \frac{1}{2}(v - v_{\mathrm{ref}})^{\mathrm{T}} W_v (v - v_{\mathrm{ref}}) + \frac{1}{2} a^{\mathrm{T}} W_a a \\ &\quad + \sum_{i \in \{\text{Swing legs}\}} \frac{1}{2}(p_i(q) - p_{i,\mathrm{ref}})^{\mathrm{T}} W_p (p_i(q) - p_{i,\mathrm{ref}}), \end{aligned} \tag{45}$$

where $a \in \mathbb{R}^{18}$ is the generalised acceleration, $q_{\mathrm{ref}}, v_{\mathrm{ref}} \in \mathbb{R}^{18}$ and $p_{i,\mathrm{ref}} \in \mathbb{R}^3$ are reference values, and $W_q, W_v, W_a \in \mathbb{R}^{18\times 18}$ and $W_p \in \mathbb{R}^{3\times 3}$ are diagonal weight matrices. q_{ref} was chosen as the generalised coordinate of the standing pose of the robot. We set the desired translational velocity of the floating base of the robot as v_{ref} and set the other element of v_{ref} as zero. The plane coordinate of $p_{i,\mathrm{ref}}$ was set as a middle point of the two successive predefined ground locations of the foot, and its vertical coordinate was set as the desired height of the swing foot. The last term in (45) changed depending on the swing legs, that is, the stage cost also switched as the active subsystem. The terminal cost $V_f(x_N)$ and impulse cost $l(x_{i_k-})$ were set as the sums of the first and second terms of (45), respectively. Similarly, the stage cost

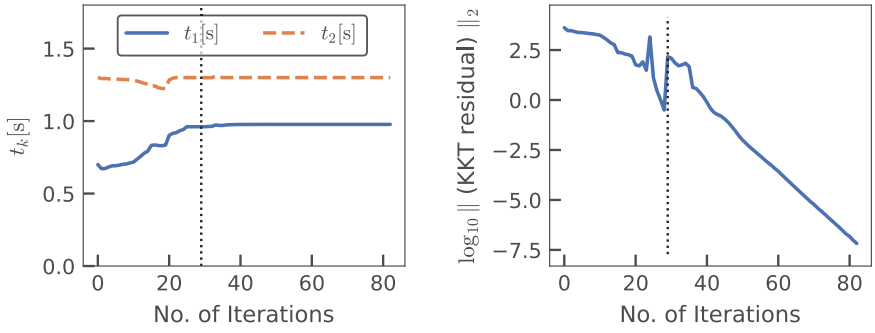

Figure 4. Convergence of the proposed method for the jumping motion of the quadrupedal robot: (a) switching instants and (b) l_2 norm of the residual in the perturbed KKT conditions over the iterations. Vertical dotted lines indicate that the mesh refinement was carried out.

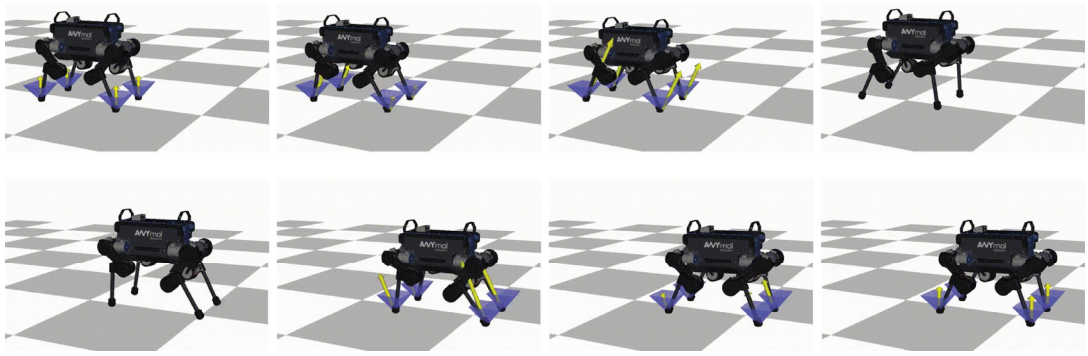

Figure 5. Snapshots of solution trajectory of the whole-body optimal control of ANYmal's 0.8 m aggressive jumping motion. The yellow arrows and blue polyhedrons represent the contact forces and friction cone constraints, respectively, which are not illustrated while the robot is flying.

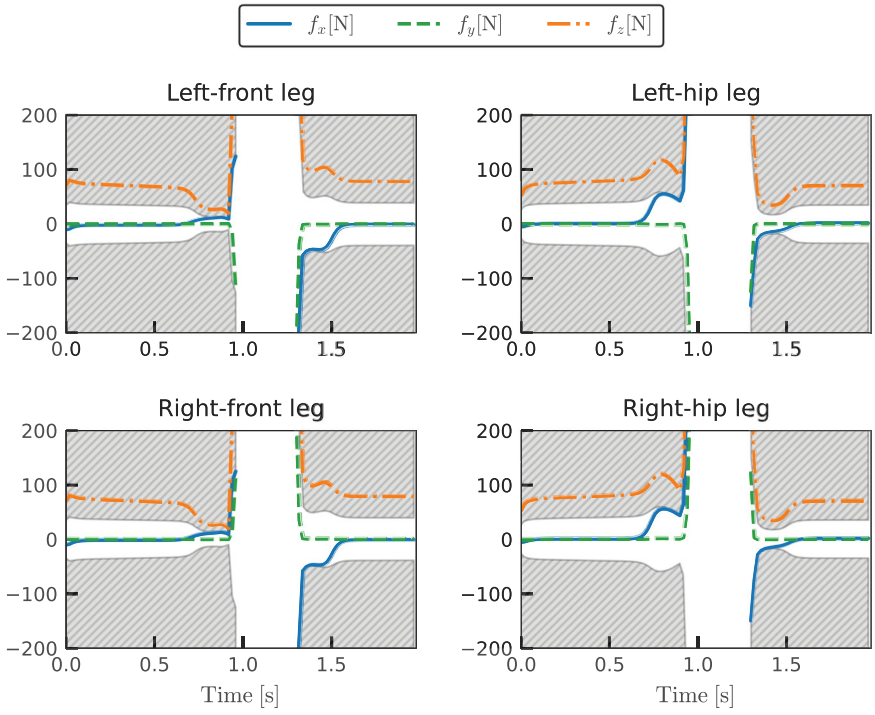

Figure 6. Time histories of the contact force expressed in the world frame $[f_x \ f_y \ f_z]$ of each leg in the jumping motion. The infeasible regions of f_x (solid lines) and f_y (dashed lines) due to the friction cone constraints are the filled gray hatches. The infeasible region of $f_z \geq 0$ (dash-dotted lines) is in the lower-half of each plot.

was designed for the jumping motions, which is expressed as

$$\frac{1}{2}(q - q_{\mathrm{ref}})^{\mathrm{T}} W_q (q - q_{\mathrm{ref}}) + \frac{1}{2} v^{\mathrm{T}} W_v v + \frac{1}{2} a^{\mathrm{T}} W_a a. \tag{46}$$

In addition, the terminal cost $V_f(x_N)$ and impulse cost $l(x_{i_k-})$ were set as the sums of the first and second terms of (46), respectively, but with different weight parameters. The minimum-dwell times in (4e) for the trotting motion were set as $\underline{\Delta}_1 = \underline{\Delta}_3 = \underline{\Delta}_5 = 0.02$ and $\underline{\Delta}_2 = \underline{\Delta}_4 = 0.2$. The minimum-dwell times were set for the jumping motion as $\underline{\Delta}_1 = \underline{\Delta}_2 = 0.2$ and $\underline{\Delta}_3 = 0.7$. A large $\underline{\Delta}_3$ was set in the jumping motion to sufficiently observe the whole-body motion after touch-down. This was because the optimiser tended to make the touch-down time very close to the end of the horizon to minimise the overall cost of the OCP without a large $\underline{\Delta}_3$.

The proposed algorithms were implemented in C++, and Eigen was used for the linear algebra. The efficient Pinocchio C++ library was used (Carpentier et al., 2019) for rigid-body dynamics (Featherstone, 2008) and its analytical derivatives (Carpentier & Mansard, 2018) to compute the dynamics and its derivatives of the quadrupedal robot. The Gauss-Newton Hessian approximation was used to avoid computing the second-order derivatives of the rigid-body dynamics. The reduced Hessian modifications (27) and (28) were used in the proposed Riccati recursion. $\sigma_{\min}$ and $\bar{\sigma}$ were chosen using (29) with $\Delta t_{k,\max} = 0.1$. Only the fraction-to-boundary rule (Nocedal & Wright, 2006; Wächter & Biegler, 2006) was used for the step-size selection. The barrier parameter was fixed at $\epsilon = 1.0 \times 10^{-3}$, which is a common suboptimal MPC setting (Wang & Boyd, 2010). OpenMP (Dagum & Menon, 1998) was used for parallel computing of the KKT system (10) in stage-wise and eight threads through the following experiments. These two experiments were conducted on a desktop computer with an octa-core CPU Intel Core i9-9900 @3.10 GHz. $\Delta\tau_{\max} = 0.02$ was used for the mesh refinement. The total number of horizon grids was fixed to $N = 50$ and $N = 90$ for the trotting and jumping problems, respectively. That is, when grids were added to the mesh-refinement phase, the same number of grids were removed from the other phases. The proposed method converged when the l_2 norm of the KKT residual was smaller than a sufficiently small threshold 1.0×10^{-7} and each $\Delta\tau_k$ was smaller than $\Delta\tau_{\max}$. Mesh refinement was performed when the l_2 norm of the KKT residual became moderately small (smaller than 0.1). After the mesh refinement, the iterates x_i, u_i, and λ_i were reconstructed by the linear interpolation of these variables that were stored before the mesh refinement. This implementation is available online as a part of our efficient optimal control solvers for robotic systems, called robotoc (Katayama, 2020).

6.2.2 Results

The convergence results of the trotting and jumping motions are shown in Figures 3 and 4, respectively. For the trotting motion that included a mesh-refinement step, the proposed method converged after 40 iterations. The total computational time was 52 ms (1.3 ms per Newton iteration). For the jumping motion that included two mesh-refinement steps, the proposed method converged after 82 iterations. The total computational time was 172 ms (2.1 ms per Newton iteration). The snapshots of the solution trajectory of the aggressive jumping motion are shown in Figure 5, and the time histories of the contact forces of the four feet in the jumping motion are shown in Figure 6. We can see that the contact forces lie inside the friction cone constraints throughout the jumping motion. These two figures show that the friction cone constraints (44) were active at time intervals immediately before and after the jumping, and the solution strictly satisfied the constraints.

The above results showed that the proposed method could achieve fast convergence and computational times per Newton iteration, even for large-scale (36-dimensional state, 12-dimensional control input) and complicated problems. In MPC, the shorter length of the horizon and smaller number of discretization grids can typically be considered, and warm-starting can be leveraged. For example, the computational time per Newton iteration for the trotting problem can be expected to be less than half of 1.3 ms if the number of grids was reduced to 20. Therefore, the proposed method is a promising approach that can achieve the whole-body MPC of robotics systems having rigid contacts within a milliseconds-range sampling time.

7. Conclusion

This study proposed an efficient Newton-type method for optimal control of switched systems under a given mode sequence. A direct multiple-shooting method with adaptive mesh refinement was formulated to guarantee the local convergence of the Newton-type method for the NLP. A dedicated structure-exploiting Riccati recursion algorithm was proposed that performed the Newton-type method for the NLP with the linear time-complexity of the total number of discretization grids. Moreover, the proposed method could always solve the KKT systems if the solution was sufficiently close to a local minimum. Conversely, general QP solvers cannot be guaranteed to solve the KKT systems because the Hessian matrix is inherently indefinite. Moreover, to enhance the convergence, a modification on the reduced Hessian matrix was proposed using the nature of the Riccati recursion algorithm as the dynamic programming for a QP subproblem. A numerical comparison was conducted with off-the-shelf solvers and demonstrated that the proposed method was up to two orders of magnitude faster. Whole-body optimal control of quadrupedal gaits was also investigated and it was demonstrated that the proposed method could achieve the whole-body MPC of robotic systems with rigid contacts.

A possible future extension of the proposed method is OCPs with free final time, including minimum-time OCPs (Bryson & Ho, 1975), because the NLP structure of such problems is expected to be similar to the proposed formulation.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was partly supported by JST SPRING, Grant Number JPMJSP2110, and JSPS KAKENHI, Grant Numbers JP22J11441 and JP22H01510.

Data availability statement

The data of 6.2 can be reproduced by our open-source software https://github.com/mayataka/robotoc. Other data that support the findings of this study are also available from the corresponding author, Sotaro Katayama, upon reasonable request.

ORCID

Toshiyuki Ohtsuka http://orcid.org/0000-0003-3554-8933

References

Amestoy, P., Duff, I. S., L'Excellent, J. Y., & Koster, J. (2001). A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, *23*(1), 15–41. https://doi.org/10.1137/S0895479899358194

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. https://doi.org/10.1007/s12532-018-0139-4

Belotti, P., Kirches, C., Leyffer, S., Linderoth, J., Luedtke, J., & Mahajan, A. (2013). Mixed-integer nonlinear optimization. *Acta Numerica*, *22*, 1–131. https://doi.org/10.1017/S0962492913000032

Bertsekas, D. P. (2005). *Dynamic programming and optimal control* (3rd ed.). Athena Scientific.

Betts, J. T. (2010). *Practical methods for optimal control and estimation using nonlinear programming* (2nd ed.). Cambridge University Press.

Bock, H., & Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problem. *9th IFAC World Congress*, *17*(2), 1603–1608. https://doi.org/10.1016/S1474-6670(17)61205-9

Bryson, A. E., & Ho, Y. C. (1975). *Applied optimal control: optimization, estimation, and control*. CRC Press.

Bürger, A., Zeile, C., Altmann-Dieses, A., Sager, S., & Diehl, M. (2019). Design, implementation and simulation of an MPC algorithm for switched nonlinear systems under combinatorial constraints. *Journal of Process Control*, *81*, 15–30. https://doi.org/10.1016/j.jprocont.2019.05.016

Carpentier, J., & Mansard, N. (2018). Analytical derivatives of rigid body dynamics algorithms. In *Robotics: science and systems (RSS 2018)* (p. hal-01790971v2f).

Carpentier, J., Saurel, G., Buondonno, G., Mirabel, J., Lamiraux, F., Stasse, O., & Mansard, N. (2019). The Pinocchio C++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *International symposium on system integration (SII)* (pp. 614–619).

Dagum, L., & Menon, R. (1998). OpenMP: an industry-standard API for shared-memory programming. *IEEE Computational Science & Engineering*, *5*(1), 46–55. https://doi.org/10.1109/99.660313

Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G., & Kim, S. (2018). Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 1–9).

Diehl, M., Bock, H. G., & Schlöder, J. P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, *43*(5), 1714–1736. https://doi.org/10.1137/S0363012902400713

Farshidian, F., Jelavic, E., Satapathy, A., Giftthaler, M., & Buchli, J (2017). Real-time motion planning of legged robots: a model predictive control approach. In *2017 IEEE-RAS 17th international conference on humanoid robotics (Humanoids)* (pp. 577–584).

Farshidian, F., Kamgarpour, M., Pardo, D., & Buchli, J. (2017). Sequential linear quadratic optimal control for nonlinear switched systems. *IFAC-PapersOnLine*, *50*(1), 1463–1469. https://doi.org/10.1016/j.ifacol.2017.08.29120th IFAC World Congress.

Featherstone, R. (2008). *Rigid body dynamics algorithms*. Springer.

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., & Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, *6*(4), 327–363. https://doi.org/10.1007/s12532-014-0071-1

Fu, J., & Zhang, C. (2021). Optimal control of path-constrained switched systems with guaranteed feasibility. *IEEE Transactions on Automatic Control*, *67*(3), 1342–1355. https://doi.org/10.1109/TAC.2021.3071035

Grandia, R., Taylor, A. J., Ames, A. D., & Hutter, M. (2021). Multi-Layered Safety for Legged Robots via Control Barrier Functions and Model Predictive Control. In *2021 IEEE international conference on robotics and automation (ICRA)* (pp. 8352–8358).

Guennebaud, G., & Jacob, B. (2010). Eigen v3. http://eigen.tuxfamily.org.

Hutter, M., Gehring, C., Lauber, A., Gunther, F., Bellicoso, C. D., Tsounis, V., Fankhauser, P., Diethelm, R., Bachmann, S., Bloesch, M., Kolvenbach, H., Bjelonic, M., Isler, L., & Meyer, K. (2017). ANYmal – toward legged robots for harsh environments. *Advanced Robotics*, *31*(17), 918–931. https://doi.org/10.1080/01691864.2017.1378591

Jenelten, F., Miki, T., Vijayan, A. E., Bjelonic, M., & Hutter, M. (2020). Perceptive locomotion in rough terrain – online foothold optimization. *IEEE Robotics and Automation Letters*, *5*, 5370–5376. https://doi.org/10.1109/LSP.2016.

Johnson, E. R., & T. D. Murphey (2011). Second-Order switching time optimization for nonlinear time-varying dynamic systems. *IEEE Transactions on Automatic Control*, *56*(8), 1953–1957. https://doi.org/10.1109/TAC.2011.2150310

Jung, M. N., Kirches, C., & Sager, S. (2013). On perspective functions and vanishing constraints in mixed-integer nonlinear optimal control. In G. R. M. Jünger (Ed.), *Facets of combinatorial optimization* (pp. 387–417). Springer.

Katayama, S. (2020). robotoc. https://github.com/mayataka/robotoc.

Katayama, S., Doi, M., & Ohtsuka, T (2020). A moving switching sequence approach for nonlinear model predictive control of switched systems with state-dependent switches and state jumps. *International Journal of Robust and Nonlinear Control*, *30*(2), 719–740. https://doi.org/10.1002/rnc.v30.2

Katayama, S., & Ohtsuka, T. (2021a). Efficient Riccati recursion for optimal control problems with pure-state equality constraints. arXiv:2102.09731.

Katayama, S., & Ohtsuka, T. (2021b). Lifted contact dynamics for efficient direct optimal control of rigid body systems with contacts. arXiv:2108.01781.

Katayama, S., & Ohtsuka, T. (2021c). Riccati recursion for optimal control problems of nonlinear switched systems. In *Proceedings of the 7th IFAC conference on nonlinear model predictive control 2021 (NMPC)* (p. 18).

Kuindersma, S., Deits, R., Fallon, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., & Tedrake, R. (2016). Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Autonomous Robots*, *40*, 429–455. https://doi.org/10.1007/s10514-015-9479-3

Li, H., & Wensing, P. M. (2020). Hybrid systems differential dynamic programming for whole-body motion planning of legged robots. *IEEE Robotics and Automation Letters*, *5*(4), 5448–5455. https://doi.org/10.1109/LSP.2016.

Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., & Semini, C. (2020). Motion planning for quadrupedal locomotion: coupled planning, terrain mapping, and whole-body control. *IEEE Transactions on Robotics*, *36*(6), 1635–1648. https://doi.org/10.1109/TRO.8860

Messerer, F., Baumgärtner, K., & Diehl, M. (2021). Survey of sequential convex programming and generalised Gauss-Newton methods. *Citation: ESAIM: Proceedings and Surveys*, *71*(1), 64–88. https://doi.org/10.1051/proc/202171107

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer.

Nurkanović, A., Albrecht, S., & Diehl, M. (2020). Limits of MPCC formulations in direct optimal control with nonsmooth differential equations. In *2020 European control conference (ECC)* (pp. 2015–2020).

Ohtsuka, T. (2004). A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, *40*(4), 563–574. https://doi.org/10.1016/j.automatica.2003.11.005

Patterson, M. A., & Rao, A. V. (2014). GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-Adaptive

Gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, *41*(1), 1–37. https://doi.org/10.1145/2558904

Posa, M., Cantu, C., & Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, *33*(1), 69–81. https://doi.org/10.1177/0278364913506757

Quirynen, R., Houska, B., Vallerio, M., Telen, D., Logist, F., Van Impe, J., & Diehl, M. (2014). Symmetric algorithmic differentiation based exact Hessian SQP method and software for Economic MPC. In *53rd IEEE conference on decision and control (CDC)* (pp. 2752–2757).

Rao, C., Wright, S. J., & Rawlings, J. B. (1998). Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, *99*(3), 723–757. https://doi.org/10.1023/A:1021711402723

Rawlings, K., Mayne, D. Q., & Diehl, M. (2017). *Model predictive control: theory, computation, and design*. Nob Hill Publishing, LCC.

Robuschi, N., Zeile, C., Sager, S., & Braghin, F. (2021). Multiphase mixed-integer nonlinear optimal control of hybrid electric vehicles. *Automatica*, *123*, 109325. https://doi.org/10.1016/j.automatica.2020.109325

Sager, S., Jung, M. N., & Kirches, C. (2011). Combinatorial integral approximation. *Mathematical Methods of Operations Research*, *73*, 363–380. https://doi.org/10.1007/s00186-011-0355-4

Schultz, G., & Mombaur, K. (2010). Modeling and optimal control of human-like running. *IEEE/ASME Transactions on Mechatronics*, *15*(5), 783–792. https://doi.org/10.1109/TMECH.2009.2035112

Sideris, A., & Rodriguez, L. A. (2011). A Riccati approach for constrained linear quadratic optimal control. *International Journal of Control*, *84*(2), 370–380. https://doi.org/10.1080/00207179.2011.555883

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. https://doi.org/10.1007/s12532-020-00179-2

Stellato, B., Ober-Blöbaum, S., & Goulart, P. J. (2017). Second-order switching time optimization for switched dynamical systems. *IEEE Transactions on Automatic Control*, *62*(10), 5407–5414. https://doi.org/10.1109/TAC.2017.2697681

Wächter, A., & Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*, 25–57. https://doi.org/10.1007/s10107-004-0559-y

Wang, Y., & Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, *18*(2), 267–278. https://doi.org/10.1109/TCST.2009.2017934

Xu, X., & Antsaklis, P. J. (2002). Optimal control of switched systems via non-linear optimization based on direct differentiations of value functions. *International Journal of Control*, *75*(16-17), 1406–1426. https://doi.org/10.1080/0020717021000023825

Xu, X., & Antsaklis, P. J. (2004). Optimal control of switched systems based on parameterization of the switching instants. *IEEE Transactions on Automatic Control*, *49*(1), 2–16. https://doi.org/10.1109/TAC.2003.821417

Yunt, K. (2011). An augmented lagrangian based shooting method for the optimal trajectory generation of switching lagrangian systems. *Dynamics of Continuous, Discrete and Impulsive Systems Series B: Applications and Algorithms*, *18*(5), 615–645. https://www.researchgate.net/profile/Kerim-Yunt/publication/255621657_An_augmented_Lagrangian_based_shooting_method_for_the_optimal_trajectory_generation_of_switching_Lagrangian_systems/links/56c9960908ae5488f0d904e7/Anaugmented-Lagrangian-based-shooting-method-for-the-optimal-trajectory-generation-of-switching-Lagrangian-systems.pdf

Zanelli, A., Domahidi, A., Jerez, J., & Morari, M. (2020). FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, *93*(1), 13–29. https://doi.org/10.1080/00207179.2017.1316017