Optimization and Engineering (2019) 20:769–809
https://doi.org/10.1007/s11081-018-9417-2

RESEARCH ARTICLE

A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)

Tobias Englert[1] · Andreas Völz[1] · Felix Mesmer[1] · Sönke Rhein[1] · Knut Graichen[1]

Received: 4 May 2018 / Revised: 1 December 2018 / Accepted: 1 December 2018 /
Published online: 18 January 2019

Abstract

A nonlinear MPC framework is presented that is suitable for dynamical systems with sampling times in the (sub)millisecond range and that allows for an efficient implementation on embedded hardware. The algorithm is based on an augmented Lagrangian formulation with a tailored gradient method for the inner minimization problem. The algorithm is implemented in the software framework GRAMPC and is a fundamental revision of an earlier version. Detailed performance results are presented for a test set of benchmark problems and in comparison to other nonlinear MPC packages. In addition, runtime results and memory requirements for GRAMPC on ECU level demonstrate its applicability on embedded hardware.

Keywords Nonlinear model predictive control · Moving horizon estimation · Augmented Lagrangian method · Gradient method · Embedded optimization · Real-time implementation

✉ Tobias Englert
tobias.englert@uni-ulm.de

Andreas Völz
andreas.voelz@fau.de

Felix Mesmer
felix.mesmer@uni-ulm.de

Sönke Rhein
soenke.rhein@alumni.uni-ulm.de

Knut Graichen
knut.graichen@fau.de

[1] Institute of Measurement, Control, and Microtechnology, Ulm University, Albert-Einstein-Allee 41, 89081 Ulm, Germany

1 Introduction

Model predictive control (MPC) is one of the most popular advanced control methods due to its ability to handle linear and nonlinear systems with constraints and multiple inputs. The well-known challenge of MPC is the numerical effort that is required to solve the underlying optimal control problem (OCP) online. In the recent past, however, the methodological as well as algorithmic development of MPC for linear and nonlinear systems has matured to a point that MPC nowadays can be applied to highly dynamical problems in real-time. Most approaches of real-time MPC either rely on suboptimal solution strategies (Scokaert et al. 1999; Diehl et al. 2004; Graichen and Kugi 2010) and/or use tailored optimization algorithms to optimize the computational efficiency.

Particularly for linear systems, the MPC problem can be reduced to a quadratic problem, for which the optimal control over the admissible polyhedral set can be precomputed. This results in an explicit MPC strategy with minimal computational effort for the online implementation (Bemporad et al. 2002a, b), though this approach is typically limited to a small number of state and control variables. An alternative to explicit MPC is the online active set method (Ferreau et al. 2008) that takes advantage of the receding horizon property of MPC in the sense that typically only a small number of constraints becomes active or inactive from one MPC step to the next.

In contrast to active set strategies, interior point methods relax the complementary conditions for the constraints and therefore solve a relaxed set of optimality conditions in the interior of the admissable constraint set. The MPC software packages FORCES (PRO) (Domahidi et al. 2012) and fast_mpc (Wang and Boyd 2010) employ interior point methods for linear MPC problems. An alternative to active set and interior point methods are accelerated gradient methods (Richter et al. 2012) that originally go back to Nesterov's fast gradient method for convex problems (Nesterov 2003). A corresponding software package for linear MPC is FiOrdOs (Jones et al. 2012). An augmented Lagrangian framework for convex quadratic problems as they arise in linear MPC is implemented in the toolbox DuQuad (Necoara and Kvamme 2015) with an analysis of the computational complexity in Nedelcu et al. (2014).

For nonlinear systems, one of the first real-time MPC algorithms was the continuation/GMRES method (Ohtsuka 2004) that solves the optimality conditions of the underlying optimal control problem based on a continuation strategy. The well-known ACADO Toolkit (Houska et al. 2011) uses the above-mentioned active set strategy in combination with a real-time iteration scheme to efficiently solve nonlinear MPC problems. Another recently presented MPC toolkit is VIATOC (Kalmari et al. 2015) that employs a projected gradient method to solve the time-discretized, linearized MPC problem.

Besides the real-time aspect of MPC, a current focus of research is on embedded MPC, i.e. the neat integration of MPC on embedded hardware with limited ressources. This might be field programmable gate arrays (FPGA) (Ling et al. 2008; Käpernick et al. 2014; Hartley and Maciejowski 2015), programmable logic controllers (PLC) in standard automation systems (Kufoalor et al. 2014; Käpernick and Graichen 2014b) or electronic control units (ECU) in automotive applications

(Mesmer et al. 2018). For embedded MPC, several challenges arise in addition to the real-time demand. For instance, numerical robustness even at low computational accuracy and tolerance against infeasibility are important aspects as well as the ability to provide fast, possibly suboptimal iterates with minimal computational demand. In this regard, it is also desirable to satisfy the system dynamics in the single MPC iterations in order to maintain dynamically consistent iterates. Low code complexity for portability and a small memory footprint are further important aspects to allow for an efficient implementation of MPC on embedded hardware. The latter aspects, in particular, require the usage of streamlined, self-contained code rather than highly complex MPC algorithms. To meet these challenges, gradient-based algorithms become popular choices due to their general simplicity and low computational complexity (Giselsson 2014; Kouzoupis et al. 2015; Necoara 2015). A comprehensive overview on literature on first-order methods for embedded optimization can be found in the survey papers (Ferreau et al. 2017; Findeisen et al. 2018) as well as in the aforementioned references (Nedelcu et al. 2014; Necoara and Kvamme 2015).

Following this motivation, this paper presents a software framework for nonlinear MPC that can be efficiently used for embedded control of nonlinear and highly dynamical systems with sampling times in the (sub)millisecond range. The presented framework is a fundamental revision of the MPC toolbox GRAMPC (Käpernick and Graichen 2014a) (Gradient-Based MPC – [græmpˈsi:]) that was originally developed for nonlinear systems with control constraints. The revised algorithm of GRAMPC presented in this paper significantly extends the applicability of the initial GRAMPC version by accounting for general nonlinear equality and inequality constraints as well as terminal constraints. Beside "classical" MPC, GRAMPC can now be used for MPC on shrinking horizon, general optimal control problems, moving horizon estimation and parameter optimization problems including free end time problems. The new algorithm employs an augmented Lagrangian formulation in connection with a real-time gradient method and tailored line search and multiplier update strategies that are optimized for a time and memory efficient implementation on embedded hardware. The performance and effectiveness of augmented Lagrangian methods for embedded nonlinear MPC was recently demonstrated for various application examples on rapid prototyping and ECU hardware level (Harder et al. 2017; Mesmer et al. 2018; Englert and Graichen 2018). Beside the presentation of the augmented Lagrangian algorithm and the general usage of GRAMPC, the paper compares its performance to the nonlinear MPC toolkits ACADO and VIATOC for different benchmark problems. Moreover, runtime results are presented for GRAMPC on dSPACE and ECU level including its memory footprint to demonstrate its applicability on embedded hardware.

The paper is organized as follows. Section 2 presents the general problem formulation and exemplarily illustrates its application to model predictive control and moving horizon estimation. Section 3 describes the augmented Lagrangian framework in combination with a gradient method for the inner minimization problem. Section 4 gives an overview on the structure and usage of GRAMPC. Section 5 evaluates the performance of GRAMPC for different benchmark problems and in comparison to ACADO and VIATOC, before Sect. 6 closes the paper.

Some norms are used inside the paper, in particular in Sect. 3. The Euclidean norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ is denoted by $\|\boldsymbol{x}\|_2$, the weighted quadratic norm by $\|\boldsymbol{x}\|_{\boldsymbol{Q}} = (\boldsymbol{x}^\mathsf{T}\boldsymbol{Q}\boldsymbol{x})^{1/2}$ for some positive definite matrix $\boldsymbol{Q}$, and the scalar product of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is defined as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\mathsf{T}\boldsymbol{y}$. For a time function $\boldsymbol{x}(t)$, $t \in [0, T]$ with $T < \infty$, the vector-valued L^2-norm is defined by $\|\boldsymbol{x}\|_{L_2} = \left(\sum_{i=1}^n \|x_i\|_{L_2} \right)^{1/2}$ with $\|x_i\|_{L_2} = \left(\int_0^T x_i^2(t)\,\mathrm{d}t \right)^{1/2}$. The supremum-norm is defined componentwise in the sense of $\|\boldsymbol{x}\|_{L_\infty} = \left[\|x_1\|_{L_\infty} \ldots \|x_n\|_{L_\infty} \right]^\mathsf{T}$ with $\|x_i\|_{L_\infty} = \sup_{t \in [0,T]} |x_i(t)|$. The inner product is denoted by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \int_0^T \boldsymbol{x}^\mathsf{T}(t)\boldsymbol{y}(t)\,\mathrm{d}t$ using the same (overloaded) $\langle\,\rangle$-notation as in the vector case. Moreover, function arguments (such as time t) might be omitted in the text for the sake of enhancing readability.

2 Problem formulation

This section describes the class of optimal control problems that can be solved by GRAMPC. The framework is especially suitable for model predictive control and moving horizon estimation, as the numerical solution method is tailored to embedded applications. Nevertheless, GRAMPC can be used to solve general optimal control problems or parameter optimization problems as well.

2.1 Optimal control problem

GRAMPC solves nonlinear constrained optimal control problems with fixed or free end time and potentially unknown parameters. Consequently, the most generic problem formulation that can be adressed by GRAMPC is given by

$$\min_{\boldsymbol{u},\boldsymbol{p},T} \quad J(\boldsymbol{u},\boldsymbol{p},T;\boldsymbol{x}_0) = V(\boldsymbol{x}(T),\boldsymbol{p},T) + \int_0^T l(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t)\,\mathrm{d}t \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{1b}$$

$$\boldsymbol{g}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) = \boldsymbol{0}, \quad \boldsymbol{g}_T(\boldsymbol{x}(T),\boldsymbol{p},T) = \boldsymbol{0} \tag{1c}$$

$$\boldsymbol{h}(\boldsymbol{x}(t),\boldsymbol{u}(t),\boldsymbol{p},t) \le \boldsymbol{0}, \quad \boldsymbol{h}_T(\boldsymbol{x}(T),\boldsymbol{p},T) \le \boldsymbol{0} \tag{1d}$$

$$\boldsymbol{u}(t) \in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right] \tag{1e}$$

$$\boldsymbol{p} \in \left[\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max}\right], \quad T \in \left[T_{\min}, T_{\max}\right] \tag{1f}$$

with state $\boldsymbol{x} \in \mathbb{R}^{N_x}$, control $\boldsymbol{u} \in \mathbb{R}^{N_u}$, parameters $\boldsymbol{p} \in \mathbb{R}^{N_p}$ and end time $T \in \mathbb{R}$. The cost to be minimized (1a) consists of the terminal and integral cost functions $V : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}$ and $l : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}$, respectively. The dynamics (1b) are given in semi-implicit form with the (constant) mass matrix $\boldsymbol{M}$, the nonlinear system function $\boldsymbol{f} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}^{N_x}$, and the initial state $\boldsymbol{x}_0$. The system class (1b) includes standard ordinary differential equations for $\boldsymbol{M} = \boldsymbol{I}$ as well as (index-1) differential-algebraic equations with singular

mass matrix $\boldsymbol{M}$. In addition, (1c) and (1d) account for equality and inequality constraints $\boldsymbol{g} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}^{N_g}$ and $\boldsymbol{h} : \mathbb{R}^{N_x} \times \mathbb{R}^{N_u} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}^{N_h}$ as well as for the corresponding terminal constraints $\boldsymbol{g}_T : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}^{N_{gT}}$ and $\boldsymbol{h}_T : \mathbb{R}^{N_x} \times \mathbb{R}^{N_p} \times \mathbb{R} \to \mathbb{R}^{N_{hT}}$, respectively. Finally, (1e) and (1f) represent box constraints for the optimization variables $\boldsymbol{u} = \boldsymbol{u}(t)$, $\boldsymbol{p}$ and T (if applicable).

In comparison to the previous version of the GRAMPC toolbox (Käpernick and Graichen 2014a), the problem formulation (1) supports optimization with respect to parameters, a free end time, general state-dependent equality and inequality constraints, as well as terminal constraints. While the original GRAMPC version was based on a real-time gradient method, the new algorithm presented in this paper employs an augmented Lagrangian framework to account for the aforementioned general constraints. Furthermore, semi-implicit dynamics with a constant mass matrix $\boldsymbol{M}$ can be handled using the Rosenbrock solver RODAS (Hairer and Wanner 1996) as numerical integrator. This extends the range of possible applications besides MPC to general optimal control, parameter optimization, and moving horizon estimation. However, the primary target is embedded model predictive control of nonlinear systems, as the numerical solution algorithm is optimized for time and memory efficiency.

2.2 Application to model predictive control

Model predictive control relies on the iterative solution of an optimal control problem of the form

$$\min_{\boldsymbol{u}} \quad J(\boldsymbol{u}; \boldsymbol{x}_k) = V(\boldsymbol{x}(T)) + \int_0^T l(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \tau)\, \mathrm{d}\tau \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(\tau) = \boldsymbol{f}(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t_k + \tau), \qquad \boldsymbol{x}(0) = \boldsymbol{x}_k \tag{2b}$$

$$\boldsymbol{x}(\tau) \in \left[\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}\right], \qquad \boldsymbol{x}(T) \in \Omega_\beta \tag{2c}$$

$$\boldsymbol{u}(\tau) \in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right] \tag{2d}$$

with the MPC-internal time coordinate $\tau \in [0, T]$ over the prediction horizon T. The initial state value $\boldsymbol{x}_k$ is the measured or estimated system state at the current sampling instant $t_k = t_0 + k\Delta t$, $k \in \mathbb{N}$ with sampling time $0 < \Delta t \leq T$. The first part of the computed control trajectory $\boldsymbol{u}(\tau)$, $\tau \in [0, \Delta t)$ is used as control input for the actual plant over the time interval $t \in [t_k, t_{k+1})$, before OCP (2) is solved again with the new initial state $\boldsymbol{x}_{k+1}$.

A popular choice of the cost functional (2b) is the quadratic form

$$V(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}}\|_{\boldsymbol{P}}^2, \quad l(\boldsymbol{x}, \boldsymbol{u}) = \|\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}}\|_{\boldsymbol{Q}}^2 + \|\boldsymbol{u} - \boldsymbol{u}_{\mathrm{des}}\|_{\boldsymbol{R}}^2 \tag{3}$$

with the desired setpoint $(\boldsymbol{x}_{\mathrm{des}}, \boldsymbol{u}_{\mathrm{des}})$ and the positive (semi-)definite matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{R}$. Stability is often ensured in MPC by imposing a terminal constraint $\boldsymbol{x}(T) \in \Omega_\beta$, where the set $\Omega_\beta = \{\boldsymbol{x} \in \mathbb{R}^{N_x} \mid V(\boldsymbol{x}) \leq \beta\}$ for some $\beta > 0$ is defined in terms of the

terminal cost $V(\boldsymbol{x})$ that can be computed from solving a Lyapunov or Riccati equation that renders the set Ω_β invariant under a local feedback law (Chen and Allgöwer 1998; Mayne et al. 2000). In view of the OCP formulation (1), the terminal region as well as general box constraints on the state as given in (2c) can be expressed as

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x}_{\max} \\ \boldsymbol{x}_{\min} - \boldsymbol{x} \end{bmatrix} \leq \boldsymbol{0}, \quad \boldsymbol{h}_T(\boldsymbol{x}) = V(\boldsymbol{x}) - \beta \leq \boldsymbol{0}. \tag{4}$$

Note, however, that terminal constraints are often omitted in embedded or real-time MPC in order to minimize the computational effort (Limon et al. 2006; Graichen et al. 2010; Grüne 2013). In particular, real-time feasibility is typically achieved by limiting the number of iterations per sampling step and using the current solution for warm starting in the next MPC step, in order to incrementally reduce the suboptimality over the runtime of the MPC (Diehl et al. 2004; Graichen et al. 2010; Graichen 2012).

An alternative to the "classical" MPC formulation (2) is shrinking horizon MPC, see e.g. Diehl et al. (2005), Skaf et al. (2010) and Grüne and Palma (2014), where the horizon length T is shortened over the MPC steps. This can be achieved by formulating the underlying OCP (2) as a free end time problem with a terminal constraint $\boldsymbol{g}_T(\boldsymbol{x}(T)) = \boldsymbol{x}(T) - \boldsymbol{x}_{\text{des}} = \boldsymbol{0}$ to ensure that a desired setpoint $\boldsymbol{x}_{\text{des}}$ is reached in finite time instead of the asymptotic behavior of fixed-horizon MPC.

2.3 Application to moving horizon estimation

Moving horizon estimation (MHE) can be seen as the dual of MPC for state estimation problems. Similar to MPC, MHE relies on the online solution of a dynamic optimization problem of the form

$$\min_{\hat{\boldsymbol{x}}_k} \quad J(\hat{\boldsymbol{x}}_k; \boldsymbol{u}, \boldsymbol{y}) = \int_{t_k - T}^{t_k} \|\hat{\boldsymbol{y}}(t) - \boldsymbol{y}(t)\|^2 \, \mathrm{d}t \tag{5a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\hat{\boldsymbol{x}}}(t) = \boldsymbol{f}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t), t), \quad \hat{\boldsymbol{x}}(t_k) = \hat{\boldsymbol{x}}_k \tag{5b}$$

$$\hat{\boldsymbol{y}}(t) = \boldsymbol{\sigma}(\hat{\boldsymbol{x}}(t)) \tag{5c}$$

that depends on the history of the control $\boldsymbol{u}(t)$ and measured output $\boldsymbol{y}(t)$ over the past time window $[t_k - T, t_k]$. The solution of (5) yields the estimate $\hat{\boldsymbol{x}}_k$ of the current state $\boldsymbol{x}_k$ such that the estimated output function (5c) best matches the measured output $\boldsymbol{y}(t)$ over the past horizon T. Further constraints can be added to the formulation of (5) to incorporate a priori knowledge.

GRAMPC can be used for moving horizon estimation by handling the system state at the beginning of the estimation horizon as optimization variables, i.e. $\boldsymbol{p} = \hat{\boldsymbol{x}}(t_k - T)$. In addition, a time transformation is required to map $t \in [t_k - T, t_k]$ to the new time coordinate $\tau \in [0, T]$ along with the corresponding coordinate transformation

$$\tilde{\boldsymbol{x}}(\tau) = \hat{\boldsymbol{x}}(t_k - T + \tau) - \boldsymbol{p}, \quad \tilde{\boldsymbol{u}}(\tau) = \boldsymbol{u}(t_k - T + \tau), \quad \tilde{\boldsymbol{y}}(\tau) = \boldsymbol{y}(t_k - T + \tau) \tag{6}$$

with the initial condition $\tilde{\boldsymbol{x}}(0) = \boldsymbol{0}$. The optimization problem (5) then becomes

$$\min_{\boldsymbol{p}} \quad J(\boldsymbol{p};\tilde{\boldsymbol{u}},\tilde{\boldsymbol{y}}) = \int_0^T \|\hat{\boldsymbol{y}}(\tau) - \tilde{\boldsymbol{y}}(\tau)\|^2 \,\mathrm{d}\tau \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\tilde{\boldsymbol{x}}}(\tau) = \boldsymbol{f}(\tilde{\boldsymbol{x}}(\tau) + \boldsymbol{p}, \tilde{\boldsymbol{u}}(\tau), t_k - T + \tau), \quad \tilde{\boldsymbol{x}}(0) = \boldsymbol{0} \tag{7b}$$

$$\hat{\boldsymbol{y}}(\tau) = \boldsymbol{\sigma}(\tilde{\boldsymbol{x}}(\tau) + \boldsymbol{p}). \tag{7c}$$

The solution $\boldsymbol{p} = \hat{\boldsymbol{x}}(t_k - T)$ of (7) and the coordinate transformation (6) are used to compute the current state estimate with

$$\hat{\boldsymbol{x}}_k = \boldsymbol{p} + \tilde{\boldsymbol{x}}(T), \tag{8}$$

where $\tilde{\boldsymbol{x}}(T)$ is the end point of the state trajectory returned by GRAMPC. In the next sampling step, the parameters can be re-initialized with the predicted estimate $\hat{\boldsymbol{x}}(t_k - T + \Delta t) = \boldsymbol{p} + \tilde{\boldsymbol{x}}(\Delta t)$.

Note that the above time transformation can alternatively be reversed in order to directly estimate the current state $\boldsymbol{p} = \hat{\boldsymbol{x}}(t_k)$. This, however, requires the reverse time integration of the dynamics, which is numerically unstable if the system is stable in forward time. Vice versa, a reverse time transformation is to be preferred for MHE of an unstable process.

3 Optimization algorithm

The optimization algorithm underlying GRAMPC uses an augmented Lagrangian formulation in combination with a real-time projected gradient method. Though SQP or interior point methods are typically superior in terms of convergence speed and accuracy, the augmented Lagrangian framework is able to rapidly provide a sub-optimal solution at low computational costs, which is important in view of real-time applications and embedded optimization. In the following, the augmented Lagrangian formulation and the corresponding optimization algorithm are described for solving OCP (1). The algorithm follows a first-optimize-then-discretize approach in order to maintain the dynamical system structure in the optimality conditions, before numerical integration is applied.

3.1 Augmented Lagrangian formulation

The basic idea of augmented Lagrangian methods is to replace the original optimization problem by its dual problem, see for example Bertsekas (1996), Nocedal and Wright (2006) and Boyd and Vandenberghe (2004) as well as Fortin and Glowinski (1983), Ito and Kunisch (1990), Bergounioux (1997), de Aguiar et al. (2016) for corresponding approaches in optimal control and function space settings.

The augmented Lagrangian formulation adjoins the constraints (1c), (1d) to the cost functional (1a) by means of multipliers $\bar{\boldsymbol{\mu}} = (\boldsymbol{\mu}_{\boldsymbol{g}}, \boldsymbol{\mu}_{\boldsymbol{h}}, \boldsymbol{\mu}_{\boldsymbol{g}_T}, \boldsymbol{\mu}_{\boldsymbol{h}_T})$ and additional quadratic penalty terms with the penalty parameters $\bar{\boldsymbol{c}} = (\boldsymbol{c}_{\boldsymbol{g}}, \boldsymbol{c}_{\boldsymbol{h}}, \boldsymbol{c}_{\boldsymbol{g}_T}, \boldsymbol{c}_{\boldsymbol{h}_T})$. A standard

approach in augmented Lagrangian theory is to transform the inequalities (1d) into equality constraints by means of slack variables, which can be analytically solved for (Bertsekas 1996). This leads to the overall set of equality constraints (see Appendix 1 for details)

$$\bar{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu}_{\boldsymbol{h}},\boldsymbol{c}_{\boldsymbol{h}}) = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t) \\ \bar{\boldsymbol{h}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu}_{\boldsymbol{h}},\boldsymbol{c}_{\boldsymbol{h}}) \end{bmatrix} = \boldsymbol{0} \tag{9a}$$

$$\bar{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_{\boldsymbol{h}_T},\boldsymbol{c}_{\boldsymbol{h}_T}) = \begin{bmatrix} \boldsymbol{g}_T(\boldsymbol{x},\boldsymbol{p},T) \\ \bar{\boldsymbol{h}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_{\boldsymbol{h}_T},\boldsymbol{c}_{\boldsymbol{h}_T}) \end{bmatrix} = \boldsymbol{0} \tag{9b}$$

with the transformed inequalities

$$\bar{\boldsymbol{h}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu}_{\boldsymbol{h}},\boldsymbol{c}_{\boldsymbol{h}}) = \max\left\{\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t), -\boldsymbol{C}_{\boldsymbol{h}}^{-1}\boldsymbol{\mu}_{\boldsymbol{h}}\right\} \tag{10a}$$

$$\bar{\boldsymbol{h}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_{\boldsymbol{h}_T},\boldsymbol{c}_{\boldsymbol{h}_T}) = \max\left\{\boldsymbol{h}_T(\boldsymbol{x},\boldsymbol{p},T), -\boldsymbol{C}_{\boldsymbol{h}_T}^{-1}\boldsymbol{\mu}_{\boldsymbol{h}_T}\right\} \tag{10b}$$

and the diagonal matrix syntax $\boldsymbol{C} = \mathrm{diag}(\boldsymbol{c})$. The vector-valued max-function is to be understood component-wise. The equalities (9a) are adjoined to the cost functional

$$\bar{J}(\boldsymbol{u},\boldsymbol{p},T,\bar{\boldsymbol{\mu}},\bar{\boldsymbol{c}};\boldsymbol{x}_0) = \bar{V}(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_T,\boldsymbol{c}_T) + \int_0^T \bar{l}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu},\boldsymbol{c})\,\mathrm{d}t \tag{11}$$

with the augmented terminal and integral cost terms

$$\begin{aligned} &\bar{V}(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_T,\boldsymbol{c}_T) \\ &\quad = V(\boldsymbol{x},\boldsymbol{p},T) + \boldsymbol{\mu}_T^{\mathsf{T}}\bar{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_{\boldsymbol{h}_T},\boldsymbol{c}_{\boldsymbol{h}_T}) + \frac{1}{2}\|\bar{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_{\boldsymbol{h}_T},\boldsymbol{c}_{\boldsymbol{h}_T})\|_{\boldsymbol{C}_T}^2 \end{aligned} \tag{12a}$$

$$\begin{aligned} &\bar{l}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu},\boldsymbol{c}) \\ &\quad = l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t) + \boldsymbol{\mu}^{\mathsf{T}}\bar{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu}_{\boldsymbol{h}},\boldsymbol{c}_{\boldsymbol{h}}) + \frac{1}{2}\|\bar{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu}_{\boldsymbol{h}},\boldsymbol{c}_{\boldsymbol{h}})\|_{\boldsymbol{C}}^2 \end{aligned} \tag{12b}$$

and the stacked penalty and multiplier vectors $\boldsymbol{\mu}_T = [\boldsymbol{\mu}_{\boldsymbol{g}_T}^{\mathsf{T}}, \boldsymbol{\mu}_{\boldsymbol{h}_T}^{\mathsf{T}}]^{\mathsf{T}}$, $\boldsymbol{c}_T = [\boldsymbol{c}_{\boldsymbol{g}_T}^{\mathsf{T}}, \boldsymbol{c}_{\boldsymbol{h}_T}^{\mathsf{T}}]^{\mathsf{T}}$, and $\boldsymbol{\mu} = [\boldsymbol{\mu}_{\boldsymbol{g}}^{\mathsf{T}}, \boldsymbol{\mu}_{\boldsymbol{h}}^{\mathsf{T}}]^{\mathsf{T}}$, $\boldsymbol{c} = [\boldsymbol{c}_{\boldsymbol{g}}^{\mathsf{T}}, \boldsymbol{c}_{\boldsymbol{h}}^{\mathsf{T}}]^{\mathsf{T}}$, respectively.

The augmented cost functional (11) allows one to formulate the max–min-problem

$$\max_{\bar{\boldsymbol{\mu}}} \min_{\boldsymbol{u},\boldsymbol{p},T} \quad \bar{J}(\boldsymbol{u},\boldsymbol{p},T,\bar{\boldsymbol{\mu}},\bar{\boldsymbol{c}};\boldsymbol{x}_0) \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)\,, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{13b}$$

$$\boldsymbol{u}(t) \in [\boldsymbol{u}_{\min},\boldsymbol{u}_{\max}]\,, \quad t \in [0,T] \tag{13c}$$

$$\boldsymbol{p} \in [\boldsymbol{p}_{\min},\boldsymbol{p}_{\max}]\,, \quad T \in [T_{\min},T_{\max}]\,. \tag{13d}$$

Note that the multipliers $\boldsymbol{\mu} = [\boldsymbol{\mu}_g^\mathsf{T}, \boldsymbol{\mu}_h^\mathsf{T}]^\mathsf{T}$ corresponding to the constraints (1d), respectively (10), are functions of time t. In the implementation of GRAMPC, the corresponding penalties $\boldsymbol{c}_g$ and $\boldsymbol{c}_h$ are handled time-dependently as well.

If strong duality holds and $(\boldsymbol{u}^*, \boldsymbol{p}^*, T^*)$ and $\bar{\boldsymbol{\mu}}^*$ are primal and dual optimal points, they form a saddle-point in the sense of

$$\bar{J}(\boldsymbol{u}^*, \boldsymbol{p}^*, T^*, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{c}}; \boldsymbol{x}_0) \leq \bar{J}(\boldsymbol{u}^*, \boldsymbol{p}^*, T^*, \bar{\boldsymbol{\mu}}^*, \bar{\boldsymbol{c}}; \boldsymbol{x}_0) \leq \bar{J}(\boldsymbol{u}, \boldsymbol{p}, T, \bar{\boldsymbol{\mu}}^*, \bar{\boldsymbol{c}}; \boldsymbol{x}_0)\,. \tag{14}$$

On the other hand, if (14) is satisfied, then $(\boldsymbol{u}^*, \boldsymbol{p}^*, T^*)$ and $\bar{\boldsymbol{\mu}}^*$ are primal and dual optimal and strong duality holds (Boyd and Vandenberghe 2004; Allaire 2007). Moreover, if the saddle-point condition is satisfied for the unaugmented Lagrangian, i.e. for $\bar{\boldsymbol{c}} = \boldsymbol{0}$, then it holds for all $\bar{\boldsymbol{c}} > \boldsymbol{0}$ and vice versa (Fortin and Glowinski 1983).

Strong duality typically relies on convexity, which is difficult to investigate for general constrained nonlinear optimization problems. However, the augmented Lagrangian formulation is favorable in this regard, as the duality gap that may occur for unpenalized, nonconvex Lagrangian formulations can potentially be closed by the augmented Lagrangian formulation (Rockafellar 1974).

The motivation behind the algorithm presented in the following lines is to solve the dual problem (13) instead of the original one (1) by approaching the saddle-point (14) from both sides. In essence, the max–min-problem (13) is solved in an alternating manner by performing the inner minimization with a projected gradient method and the outer maximization via a steepest ascent approach. Note that the dynamics (13b) are captured inside the minimization problem instead of treating the dynamics as equality constraints of the form $\boldsymbol{M}\dot{\boldsymbol{x}} - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) = \boldsymbol{0}$ in the augmented Lagrangian (Hager 1990). This ensures the dynamical consistency of the computed trajectories in each iteration of the algorithm, which is important for an embedded, possibly suboptimal implementation.

3.2 Structure of the augmented Lagrangian algorithm

The basic iteration structure of the augmented Lagrangian algorithm is summarized in Algorithm 1 and will be detailed in Sects. 3.3–3.5. The initialization of the algorithm concerns the multipliers $\bar{\boldsymbol{\mu}}^1$ and penalties $\bar{\boldsymbol{c}}^1$ as well as the definition of several tolerance values that are used for the convergence check (Sect. 3.4) and the update of the multipliers and penalties in (17) and (18), respectively.

In the current augmented Lagrangian iteration i, the inner minimization is carried out by solving the OCP (15) for the current set of multipliers $\bar{\boldsymbol{\mu}}^i$ and penalties $\bar{\boldsymbol{c}}^i$. Since the only remaining constraints within (15) are box constraints on the optimization variables (15c) and (15d), the problem can be efficiently solved by the projected gradient method described in Sect. 3.3. The solution of the minimization step consists of the control vector $\boldsymbol{u}^i$ and of the parameters $\boldsymbol{p}^i$ and free end time T^i, if these are specified in the problem at hand. The subsequent convergence check in

Algorithm 1 Augmented Lagrangian algorithm

Initialization

- Initialize multipliers $\bar{\boldsymbol{\mu}}^0$ and penalties $\bar{\boldsymbol{c}}^0$
- Set tolerances ($\boldsymbol{\varepsilon}_{\boldsymbol{g}} > 0$, $\boldsymbol{\varepsilon}_{\boldsymbol{h}} > 0$, $\boldsymbol{\varepsilon}_{\boldsymbol{g}_T} > 0$, $\boldsymbol{\varepsilon}_{\boldsymbol{h}_T} > 0$, $\varepsilon_{\mathrm{rel,c}} > 0$)

for $i = 1$ **to** $i_{\max}$ **do**

- Compute $(\boldsymbol{u}^i, \boldsymbol{p}^i, T^i)$ with $\boldsymbol{u}^i = \boldsymbol{u}^i(t)$ and state $\boldsymbol{x}^i = \boldsymbol{x}^i(t)$, $t \in [0, T^i]$ by solving the optimal control problem (see Section 3.3)

$$\min_{\boldsymbol{u},\boldsymbol{p},T} \quad \bar{J}(\boldsymbol{u}, \boldsymbol{p}, T, \bar{\boldsymbol{\mu}}^i, \bar{\boldsymbol{c}}^i; \boldsymbol{x}_0) \tag{15a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t), \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{15b}$$

$$\boldsymbol{u}(t) \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}], \quad t \in [0, T] \tag{15c}$$

$$\boldsymbol{p} \in [\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max}], \quad T \in [T_{\min}, T_{\max}] \tag{15d}$$

- Store constraint functions

$$\boldsymbol{g}^i(t) = \boldsymbol{g}(\boldsymbol{x}^i(t), \boldsymbol{u}^i(t), t), \qquad \boldsymbol{g}_T^i = \boldsymbol{g}_T(\boldsymbol{x}^i(T), T^i) \tag{16a}$$

$$\bar{\boldsymbol{h}}^i(t) = \bar{\boldsymbol{h}}(\boldsymbol{x}^i(t), \boldsymbol{u}^i(t), \boldsymbol{p}^i, t, \boldsymbol{\mu}_{\boldsymbol{h}}^i(t), \boldsymbol{c}_{\boldsymbol{h}}^i(t)), \quad \boldsymbol{h}_T^i = \bar{\boldsymbol{h}}_T(\boldsymbol{x}^i(T), \boldsymbol{p}^i, T^i, \boldsymbol{\mu}_{\boldsymbol{h}_T}^i, \boldsymbol{c}_{\boldsymbol{h}_T}^i) \tag{16b}$$

- If convergence criterion is reached (see Section 3.4), **break**
- Update multipliers $\bar{\boldsymbol{\mu}}^i = (\boldsymbol{\mu}_{\boldsymbol{g}}^i, \boldsymbol{\mu}_{\boldsymbol{h}}^i, \boldsymbol{\mu}_{\boldsymbol{g}_T}^i, \boldsymbol{\mu}_{\boldsymbol{h}_T}^i)$ according to (see Section 3.5)

$$\boldsymbol{\mu}_{\boldsymbol{g}}^{i+1}(t) = \boldsymbol{\zeta}_{\boldsymbol{g}}\left(\boldsymbol{\mu}_{\boldsymbol{g}}^i(t), \boldsymbol{c}_{\boldsymbol{g}}^i(t), \boldsymbol{g}^i(t), \boldsymbol{\varepsilon}_{\boldsymbol{g}}\right), \quad \boldsymbol{\mu}_{\boldsymbol{g}_T}^{i+1} = \boldsymbol{\zeta}_{\boldsymbol{g}_T}\left(\boldsymbol{\mu}_{\boldsymbol{g}_T}^i, \boldsymbol{c}_{\boldsymbol{g}_T}^i, \boldsymbol{g}_T^i, \boldsymbol{\varepsilon}_{\boldsymbol{g}_T}\right) \tag{17a}$$

$$\boldsymbol{\mu}_{\boldsymbol{h}}^{i+1}(t) = \boldsymbol{\zeta}_{\boldsymbol{h}}\left(\boldsymbol{\mu}_{\boldsymbol{h}}^i(t), \boldsymbol{c}_{\boldsymbol{h}}^i(t), \bar{\boldsymbol{h}}^i(t), \boldsymbol{\varepsilon}_{\boldsymbol{h}}\right), \quad \boldsymbol{\mu}_{\boldsymbol{h}_T}^{i+1} = \boldsymbol{\zeta}_{\boldsymbol{h}_T}\left(\boldsymbol{\mu}_{\boldsymbol{h}_T}^i, \boldsymbol{c}_{\boldsymbol{h}_T}^i, \bar{\boldsymbol{h}}_T^i, \boldsymbol{\varepsilon}_{\boldsymbol{h}_T}\right) \tag{17b}$$

- Update penalties $\bar{\boldsymbol{c}}^i = (\boldsymbol{c}_{\boldsymbol{g}}^i, \boldsymbol{c}_{\boldsymbol{h}}^i, \boldsymbol{c}_{\boldsymbol{g}_T}^i, \boldsymbol{c}_{\boldsymbol{h}_T}^i)$, if $i \geq 2$ or MPC warmstart (see Section 3.5)

$$\boldsymbol{c}_{\boldsymbol{g}}^{i+1}(t) = \boldsymbol{\xi}_{\boldsymbol{g}}\left(\boldsymbol{c}_{\boldsymbol{g}}^i(t), \boldsymbol{g}^i(t), \boldsymbol{g}^{i-1}(t), \boldsymbol{\varepsilon}_{\boldsymbol{g}}\right), \quad \boldsymbol{c}_{\boldsymbol{g}_T}^{i+1} = \boldsymbol{\xi}_{\boldsymbol{g}_T}\left(\boldsymbol{c}_{\boldsymbol{g}_T}^i, \boldsymbol{g}_T^i, \boldsymbol{g}_T^{i-1}, \boldsymbol{\varepsilon}_{\boldsymbol{g}_T}\right) \tag{18a}$$

$$\boldsymbol{c}_{\boldsymbol{h}}^{i+1}(t) = \boldsymbol{\xi}_{\boldsymbol{h}}\left(\boldsymbol{c}_{\boldsymbol{h}}^i(t), \bar{\boldsymbol{h}}^i(t), \bar{\boldsymbol{h}}^{i-1}(t), \boldsymbol{\varepsilon}_{\boldsymbol{h}}\right), \quad \boldsymbol{c}_{\boldsymbol{h}_T}^{i+1} = \boldsymbol{\xi}_{\boldsymbol{h}_T}\left(\boldsymbol{c}_{\boldsymbol{h}_T}^i, \bar{\boldsymbol{h}}_T^i, \bar{\boldsymbol{h}}_T^{i-1}, \boldsymbol{\varepsilon}_{\boldsymbol{h}_T}\right) \tag{18b}$$

end

Algorithm 1 rates the constraint violation as well as convergence behavior of the previous minimization step and is detailed in Sect. 3.4.

If convergence is not reached yet, the multipliers and penalties are updated for the next iteration of the algorithm, as detailed in Sect. 3.5. Note that the penalty update in (18) relies on the last two iterates of the constraint functions (16). In the initial iteration $i = 1$ and if GRAMPC is used within an MPC setting, the constraint functions $\boldsymbol{g}^0$, $\boldsymbol{h}^0$, $\boldsymbol{g}_T^0$, $\boldsymbol{h}_T^0$ are warm-started by the corresponding last iterations of the previous MPC run. Otherwise, the penalty update is started in iteration $i = 2$.

Note that if the OCP or MPC problem to be solved is defined without the nonlinear constraints (1c) and (1d), the overall augmented Lagrangian algorithm reduces to the projected gradient method for solving the minimization problem (15), for which linear convergence results exist under the assumption of convexity (Dunn 1996).

If the end time T is treated as optimization variable as shown in Algorithm 1, the evaluation of (18) would formally require to redefine the constraint functions $\boldsymbol{g}^{i-1}(t)$ and $\boldsymbol{h}^{i-1}(t)$, $t \in [0, T^{i-1}]$ from the previous iterations to the new horizon length T^i by either shrinkage or extension. This redefinition is not explicitly stated in Algorithm 1, since the actual implementation of GRAMPC stores the trajectories in discretized form, which implies that only the discretized time vector must be recomputed, once the end time T^i is updated.

3.3 Gradient algorithm for inner minimization problem

The OCP (15) inside the augmented Lagrangian algorithm corresponds to the inner minimization problem of the max–min-formulation (13) for the current iterates of the multipliers $\bar{\boldsymbol{\mu}}^i$ and $\bar{\boldsymbol{c}}^i$. A projected gradient method is used to solve OCP (15) to a desired accuracy or for a fixed number of iterations.

The gradient algorithm relies on the solution of the first-order optimality conditions defined in terms of the Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, \lambda, t, \boldsymbol{\mu}, \boldsymbol{c}) = \bar{l}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t, \boldsymbol{\mu}, \boldsymbol{c}) + \lambda^{\mathrm{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t) \tag{19}$$

with the adjoint states $\lambda \in \mathbb{R}^{N_x}$. In particular, the gradient algorithm iteratively solves the canonical equations, see e.g. Cao et al. (2003),

$$\boldsymbol{M}\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, t)\,, \qquad \boldsymbol{x}(0) = \boldsymbol{x}_0, \tag{20a}$$

$$\boldsymbol{M}^{\mathrm{T}} \dot{\lambda} = -H_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}, \lambda, t, \boldsymbol{\mu}, \boldsymbol{c})\,, \qquad \boldsymbol{M}^{\mathrm{T}} \lambda(T) = \bar{V}_{\boldsymbol{x}}(\boldsymbol{x}(T), \boldsymbol{p}, T, \boldsymbol{\mu}_T, \boldsymbol{c}_T) \tag{20b}$$

consisting of the original dynamics (20a) and the adjoint dynamics (20b) in forward and backward time and computes a gradient update for the control in order to minimize the Hamiltonian in correspondence with Pontryagin's Maximum Principle (Kirk 1970; Berkovitz 1974), i.e.

$$\min_{\boldsymbol{u} \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]} H\big(\boldsymbol{x}(t), \boldsymbol{u}, \boldsymbol{p}, \lambda(t), t, \boldsymbol{\mu}(t), \boldsymbol{c}(t)\big)\,, \quad t \in [0, T]\,. \tag{21}$$

If parameters $\boldsymbol{p}$ and/or the end time T are additional optimization variables, the corresponding gradients have to be computed as well. Algorithm 2 lists the overall projected gradient algorithm for the full optimization case, i.e. for the optimization variables $(\boldsymbol{u}, \boldsymbol{p}, T)$, for the sake of completeness.

Algorithm 2 Augmented Lagrangian algorithm

Initialization

- Initialize $\boldsymbol{u}^{i|1}(t)$, $t \in [0, T^{i|1}]$ and $\boldsymbol{p}^{i|1}$, $T^{i|1}$
- Compute $\boldsymbol{x}^{i|1}(t)$, $t \in [0, T^{i|1}]$ by solving (26) for $j = 0$
- Set step size adaptation factors $\gamma_{\boldsymbol{p}} > 0$, $\gamma_T > 0$

for $j = 1$ **to** $j_{\max}$ **do**

- Compute $\boldsymbol{\lambda}^{i|j}(t)$ by backward integration of

$$\boldsymbol{M}^{\mathsf{T}}\dot{\boldsymbol{\lambda}}^{i|j}(t) = -H_{\boldsymbol{x}}\left(\boldsymbol{x}^{i|j}(t), \boldsymbol{u}^{i|j}(t), \boldsymbol{p}^{i|j}, \boldsymbol{\lambda}^{i|j}(t), t, \boldsymbol{\mu}^{i}(t), \boldsymbol{c}^{i}(t)\right),$$
$$\boldsymbol{M}^{\mathsf{T}}\boldsymbol{\lambda}^{i|j}(T^{i|j}) = \bar{V}_{\boldsymbol{x}}\left(\boldsymbol{x}^{i|j}(T^{i|j}), T^{i|j}, \boldsymbol{\mu}_T^{i}, \boldsymbol{c}_T^{i}\right) \tag{22}$$

- Compute gradients with $H^{i|j}(t) := H\left(\boldsymbol{x}^{i|j}(t), \boldsymbol{u}^{i|j}(t), \boldsymbol{p}^{i|j}, \boldsymbol{\lambda}^{i|j}(t), t, \boldsymbol{\mu}^{i}(t), \boldsymbol{c}^{i}(t)\right)$

$$\boldsymbol{d}_{\boldsymbol{u}}^{i|j}(t) = H_{\boldsymbol{u}}\left(\boldsymbol{x}^{i|j}(t), \boldsymbol{u}^{i|j}(t), \boldsymbol{p}^{i|j}, \boldsymbol{\lambda}^{i|j}(t), t, \boldsymbol{\mu}^{i}(t), \boldsymbol{c}^{i}(t)\right) \tag{23a}$$
$$\boldsymbol{d}_{\boldsymbol{p}}^{i|j} = \bar{V}_{\boldsymbol{p}}(\boldsymbol{x}^{i|j}(T^{i|j}), \boldsymbol{p}^{i|j}, T^{i|j}, \boldsymbol{\mu}_T^{i}, \boldsymbol{c}_T^{i}) + \int_0^{T^{i|j}} H_{\boldsymbol{p}}^{i|j}(t)\,\mathrm{d}t \tag{23b}$$
$$d_T^{i|j} = \bar{V}_T(\boldsymbol{x}^{i|j}(T^{i|j}), \boldsymbol{p}^{i|j}, T^{i|j}, \boldsymbol{\mu}_T^{i}, \boldsymbol{c}_T^{i}) + H^{i|j}(T^{i|j}) \tag{23c}$$

- Compute step size $\alpha^{i|j}$ by solving the line search problem

$$\min_{\alpha>0}\ \bar{J}\left(\boldsymbol{\psi}_{\boldsymbol{u}}\left(\boldsymbol{u}^{i|j} - \alpha\,\boldsymbol{d}_{\boldsymbol{u}}^{i|j}\right), \boldsymbol{\psi}_{\boldsymbol{p}}\left(\boldsymbol{p}^{i|j} - \gamma_{\boldsymbol{p}}\,\alpha\,\boldsymbol{d}_{\boldsymbol{p}}^{i|j}\right), \psi_T\left(T^{i|j} - \gamma_T\,\alpha\,\boldsymbol{d}_T^{i|j}\right); \boldsymbol{x}_0\right) \tag{24}$$

- Update $(\boldsymbol{u}^{i|j+1}, \boldsymbol{p}^{i|j+1}, T^{i|j+1})$ according to

$$\boldsymbol{u}^{i|j+1}(t) = \boldsymbol{\psi}_{\boldsymbol{u}}\left(\boldsymbol{u}^{i|j}(t) - \alpha^{i|j}\boldsymbol{d}_{\boldsymbol{u}}^{i|j}(t)\right) \tag{25a}$$
$$\boldsymbol{p}^{i|j+1} = \boldsymbol{\psi}_{\boldsymbol{p}}\left(\boldsymbol{p}^{i|j} - \gamma_{\boldsymbol{p}}\,\alpha^{i|j}\boldsymbol{d}_{\boldsymbol{p}}^{i|j}\right) \tag{25b}$$
$$T^{i|j+1} = \psi_T\left(T^{i|j} - \gamma_T\,\alpha^{i|j} d_T^{i|j}\right) \tag{25c}$$

- Compute $\boldsymbol{x}^{i|j+1}(t)$ by forward integration of

$$\boldsymbol{M}\dot{\boldsymbol{x}}^{i|j+1}(t) = \boldsymbol{f}\left(\boldsymbol{x}^{i|j+1}(t), \boldsymbol{u}^{i|j+1}(t), \boldsymbol{p}^{i|j+1}, t\right), \quad \boldsymbol{x}^{i|j+1}(0) = \boldsymbol{x}_0 \tag{26}$$

- Evaluate convergence criterion

$$\eta^{i|j+1} = \max\left\{\frac{\|\boldsymbol{u}^{i|j+1} - \boldsymbol{u}^{i|j}\|_{L_2}}{\|\boldsymbol{u}^{i|j+1}\|_{L_2}}, \frac{\left\|\boldsymbol{p}^{i|j+1} - \boldsymbol{p}^{i|j}\right\|_2}{\left\|\boldsymbol{p}^{i|j+1}\right\|_2}, \frac{|T^{i|j+1} - T^{i|j}|}{T^{i|j+1}}\right\} \tag{27}$$

- If $\eta^{i|j+1} \leq \varepsilon_{\text{rel,c}}$ or $j = j_{\max}$, **break**

end

Output to Algorithm 1

- Set $\boldsymbol{u}^{i}(t) := \boldsymbol{u}^{i|j+1}(t)$ and $\boldsymbol{x}^{i}(t) := \boldsymbol{x}^{i|j+1}(t)$
- Set $\boldsymbol{p}^{i} := \boldsymbol{p}^{i|j+1}$ and $T^{i} := T^{i|j+1}$
- Return $\eta^{i} := \eta^{i|j+1}$

The gradient algorithm is initialized with an initial control $\boldsymbol{u}^{i|1}(t)$ and initial parameters $\boldsymbol{p}^{i|1}$ and time length $T^{i|1}$. In case of MPC, these initial values are taken from the last sampling step using a warmstart strategy with an optional time shift in order to compensate for the horizon shift by the sampling time Δt.

The algorithm starts in iteration $j = 1$ with computing the corresponding state trajectory $\boldsymbol{x}^{i|j}(t)$ as well as the adjoint state trajectory $\boldsymbol{\lambda}^{i|j}(t)$ by integrating the adjoint dynamics (22) in reverse time. More detailed information on the integration schemes can be found in the GRAMPC manual. In the next step, the gradients (23) are computed and the step size $\alpha^{i|j} > 0$ is determined from the line search problem (24). The projection functions $\boldsymbol{\psi}_{\boldsymbol{u}}(\boldsymbol{u})$, $\boldsymbol{\psi}_{\boldsymbol{p}}(\boldsymbol{p})$, and $\psi_T(T)$ project the inputs, the parameters, and the end time onto the feasible sets (15c) and (15d). For instance, $\boldsymbol{\psi}_{\boldsymbol{u}}(\boldsymbol{u}) = [\psi_{\boldsymbol{u},1}(u_1) \dots \psi_{\boldsymbol{u},N_{\boldsymbol{u}}}(u_{N_{\boldsymbol{u}}})]^\mathrm{T}$ is defined by

$$\psi_{\boldsymbol{u},i}(u_i) = \begin{cases} u_i & \text{if } u_i \in (u_{\min,i}, u_{\max,i}) \\ u_{\min,i} & \text{if } u_i \le u_{\min,i}\,, \\ u_{\max,i} & \text{if } u_i \ge u_{\max,i} \end{cases} \qquad i = 1, \dots, N_{\boldsymbol{u}}\,. \tag{28}$$

The next steps in Algorithm 2 are the updates (25) of the control $\boldsymbol{u}^{i|j+1}$, parameters $\boldsymbol{p}^{i|j+1}$, and end time $T^{i|j+1}$ as well as the update of the state trajectory $\boldsymbol{x}^{i|j+1}$ in (26).

The convergence measure $\eta^{i|j+1}$ in (27) rates the relative gradient changes of $\boldsymbol{u}$, $\boldsymbol{p}$, and T. If the gradient scheme converges in the sense of $\eta^{i|j+1} \le \varepsilon_{\mathrm{rel,c}}$ with threshold $\varepsilon_{\mathrm{rel,c}} > 0$ or if the maximum number of iterations $j_{\max}$ is reached, the algorithm terminates and returns the last solution to Algorithm 1. Otherwise, j is incremented and the gradient iteration continues.

An important component of Algorithm 2 is the line search problem (24), which is performed in all search directions simultaneously. The scaling factors $\gamma_{\boldsymbol{p}}$ and γ_T can be used to scale the step sizes relative to each other, if they are not of the same order of magnitude or if the parameter or end time optimization is highly sensitive. GRAMPC implements two different line search strategies, an adaptive and an explicit one, in order to solve (24) in an accurate and robust manner without involving too much computational load.

The adaptive strategy evaluates the cost functional (24) for three different step sizes, i.e. $(\alpha_i, \bar{J}_i)$, $i = 1, 2, 3$ with $\alpha_1 < \alpha_2 < \alpha_3$, in order to compute a polynomial fitting function $\Phi(\alpha)$ of the form

$$\Phi(\alpha) = p_0 + p_1\alpha + p_2\alpha^2\,, \tag{29}$$

where the constants p_i are computed from the test points $(\alpha_i, \bar{J}_i)$, $i = 1, 2, 3$. The approximate step size can then be analytically derived by solving

$$\alpha^{i|j} = \underset{\alpha \in [\alpha_1, \alpha_3]}{\arg\min}\ \Phi(\alpha)\,. \tag{30}$$

The interval $[\alpha_1, \alpha_3]$ is adapted in the next gradient iteration, if the step size α^j is close to the interval's borders, see Käpernick and Graichen (2014a) for more details.

Depending on the OCP or MPC problem at hand, the adaptive line search method may not be suited for time-critical applications, since the approximation of the cost function (29) requires to integrate the system dynamics (15b) and the cost integral (15a) three times. This computational load can be further reduced by an explicit line search strategy, originally discussed in Barzilai and Borwein (1988) and adapted to the optimal control case in Käpernick and Graichen (2013). Motivated by the secant equation in quasi-Newton methods (Barzilai and Borwein 1988), this strategy minimizes the difference between two updates of the optimization variables

$(\boldsymbol{u}, \boldsymbol{p}, T)$ for the same step size and without considering the corresponding box constraints (15c) and (15d). The explicit method solves the problem

$$\begin{aligned}&\min_{\alpha>0} \left\|\boldsymbol{u}^{i\vert j+1} - \boldsymbol{u}^{i\vert j}\right\|_{L_2}^2 + \left\|\boldsymbol{p}^{i\vert j+1} - \boldsymbol{p}^{i\vert j}\right\|_2^2 + \left(T^{i\vert j+1} - T^{i\vert j}\right)^2 \\ &\quad = \min_{\alpha>0} \left\|\Delta\boldsymbol{u}^{i\vert j} - \alpha\Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}\right\|_{L_2}^2 + \left\|\Delta\boldsymbol{p}^{i\vert j} - \gamma_{\boldsymbol{p}}\alpha\Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}\right\|_2^2 + \left\|\Delta T^{i\vert j} - \gamma_T\alpha\Delta d_T^{i\vert j}\right\|_2^2 .\end{aligned} \tag{31}$$

where Δ denotes the difference between the last and current iterate, e.g. $\Delta\boldsymbol{u}^{i\vert j} = \boldsymbol{u}^{i\vert j} - \boldsymbol{u}^{i\vert j-1}$. The analytic solution is given by

$$\alpha^{i\vert j} = \frac{\langle\Delta\boldsymbol{u}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}\rangle + \gamma_{\boldsymbol{p}}\langle\Delta\boldsymbol{p}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}\rangle + \gamma_T\Delta T^{i\vert j}\Delta d_T^{i\vert j}}{\langle\Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}\rangle + \gamma_{\boldsymbol{p}}^2\langle\Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}\rangle + \gamma_T^2\left(\Delta d_T^{i\vert j}\right)^2} . \tag{32}$$

Alternatively, the minimization

$$\min_{\alpha>0} \left\|\alpha\Delta\boldsymbol{u}^{i\vert j} - \Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}\right\|_{L_2}^2 + \left\|\gamma_{\boldsymbol{p}}\alpha\Delta\boldsymbol{p}^{i\vert j} - \Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}\right\|_2^2 + \left(\gamma_T\alpha\Delta T^{i\vert j} - \Delta d_T^{i\vert j}\right)^2$$

can be carried out, similar to (31), leading to the corresponding solution

$$\alpha^{i\vert j} = \frac{\langle\Delta\boldsymbol{u}^{i\vert j}, \Delta\boldsymbol{u}^{i\vert j}\rangle + \gamma_{\boldsymbol{p}}\langle\Delta\boldsymbol{p}^{i\vert j}, \Delta\boldsymbol{p}^{i\vert j}\rangle + \gamma_T\left(\Delta T^{i\vert j}\right)^2}{\langle\Delta\boldsymbol{u}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{u}}^{i\vert j}\rangle + \gamma_{\boldsymbol{p}}^2\langle\Delta\boldsymbol{p}^{i\vert j}, \Delta\boldsymbol{d}_{\boldsymbol{p}}^{i\vert j}\rangle + \gamma_T^2\Delta T^{i\vert j}\Delta d_T^{i\vert j}} . \tag{33}$$

Both explicit formulas (32) and (33) are implemented in GRAMPC as an alternative to the adaptive line search strategy.

3.4 Convergence criterion

The gradient scheme in Algorithm 2 solves the inner minimization problem (15) of the augmented Lagrangian algorithm and returns the solution $(\boldsymbol{u}^i, \boldsymbol{p}^i, T^i)$ as well as the maximum relative gradient η^i that is computed in (27) and used to check convergence inside the gradient algorithm. The outer augmented Lagrangian iteration in Algorithm 1 also uses this criterion along with the convergence check of the constraints, i.e.

$$\begin{bmatrix} \vert \boldsymbol{g}_T^i \vert \\ \max\{\bar{\boldsymbol{h}}_T^i, \boldsymbol{0}\} \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\varepsilon}_{g_T} \\ \boldsymbol{\varepsilon}_{h_T} \end{bmatrix} \quad \wedge \quad \max_{t\in[0,T]} \begin{bmatrix} \vert \boldsymbol{g}^i(t) \vert \\ \max\{\bar{\boldsymbol{h}}^i(t), \boldsymbol{0}\} \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\varepsilon}_{\boldsymbol{g}} \\ \boldsymbol{\varepsilon}_{\boldsymbol{h}} \end{bmatrix} \quad \wedge \quad \eta^i \leq \varepsilon_{\text{rel,c}} . \tag{34}$$

The thresholds $\boldsymbol{\varepsilon}_{\boldsymbol{g}}, \boldsymbol{\varepsilon}_{g_T}, \boldsymbol{\varepsilon}_{\boldsymbol{h}}, \boldsymbol{\varepsilon}_{h_T}$ are vector-valued to rate each constraint individually. If the maximum number of augmented Lagrangian iterations is reached, i.e. $i = i_{\max}$, the algorithm terminates in order to ensure real-time feasibility. Otherwise, i is incremented and the next minimization of problem (15) is carried out.

Note that the convergence criterion is typically deactivated in MPC applications and a fixed iteration count is used to ensure real-time feasibility. The last solution is then used as warm-start in the next MPC step. The incremental improvement of this suboptimal solution strategy in a real-time MPC setting is investigated e.g. in Diehl et al. (2004) and

Graichen and Kugi (2010). In particular, it is shown in Graichen and Kugi (2010) that incremental improvement and asymtptotic stability is ensured for a sufficient number of iterations provided that the underlying optimization is linearly convergent.

If the convergence criterion is replaced by a fixed number of iterations, convergence of the augmented Lagrangian algorithm cannot be guaranteed in general, since the saddle-point condition (14) is based on the optimal solution of the inner problem. To remedy this issue, the multiplier update is adapted accordingly, which is explained in detail in the next section.

3.5 Update of multipliers and penalties

The multiplier update (17) in Algorithm 1 is carried out via a steepest ascent approach, whereas the penalty update (18) uses an adaptation strategy that rates the progress of the last two iterates. In more detail, the multiplier update function for a single equality constraint $g^i := g(\boldsymbol{x}^i, \boldsymbol{u}^i, \boldsymbol{p}^i, t)$ is defined by

$$\zeta_g(\mu_g^i, c_g^i, g^i, \varepsilon_g) = \begin{cases} \mu_g^i + (1-\rho) c_g^i g^i & \text{if } |g^i| > \varepsilon_g \ \wedge \ \eta^i \leq \varepsilon_{\text{rel,u}} \\ \mu_g^i & \text{else}\,. \end{cases} \tag{35}$$

The steepest ascent direction is the residual of the constraint g^i with the penalty c_g^i serving as step size parameter. The additional damping factor $0 \leq \rho \leq 1$ is introduced to increase the robustness of the multiplier update in the augmented Lagrangian scheme. In the GRAMPC implementation, the multipliers are additionally limited by an upper bound $\mu_{\max}$, in order to avoid unlimited growth and numerical stability issues. The update (35) is skipped if the constraint is satisfied within the tolerance ε_g or if the gradient method is not sufficiently converged, which is checked by the maximum relative gradient η^i, cf. (27), and the update threshold $\varepsilon_{\text{rel,u}} > 0$. GRAMPC uses a larger value for $\varepsilon_{\text{rel,u}}$ than the convergence threshold $\varepsilon_{\text{rel,c}}$ of the gradient scheme in Algorithm 2. This accounts for the case that the gradient algorithm might not have converged to the desired tolerance before the maximum number of iterations $j_{\max}$ is reached, e.g. in real-time MPC applications, where only one or two gradient iterations might be applied. In this case, $\varepsilon_{\text{rel,u}} \gg \varepsilon_{\text{rel,c}}$ ensures that the multiplier update is still performed provided that at least "some" convergence was reached by the inner minimization.

The penalty c_g^i corresponding to the equality constraint g^i is updated according to the heuristic update function

$$\xi_g(c_g^i, g^i, g^{i-1}, \varepsilon_g) = \begin{cases} \beta_{\text{in}} c_g^i & \text{if } |g^i| \geq \max\left\{\gamma_{\text{in}} |g^{i-1}|, \varepsilon_g\right\} \wedge \eta^i \leq \varepsilon_{\text{rel,u}} \\ \beta_{\text{de}}\, c_g^i & \text{else if } |g^i| \leq \gamma_{\text{de}}\, \varepsilon_g \\ c_g^i & \text{else} \end{cases} \tag{36}$$

that is basically motivated by the LANCELOT package (Conn et al. 1992; Nocedal and Wright 2006). The penalty c_g^i is increased by the factor $\beta_{\text{in}} \geq 1$, if the last gradient scheme converged, i.e. $\eta^i \leq \varepsilon_{\text{rel,u}}$, but insufficient progress (rated by $\gamma_{\text{in}} > 0$) was made by the constraint violation compared to the previous iteration $i-1$. The penalty is decreased by the factor $\beta_{\text{de}} \leq 1$ if the constraint g^i is sufficiently satisfied within its

tolerance with $0 < \gamma_{\text{de}} < 1$. The setting $\beta_{\text{de}} = \beta_{\text{in}} = 1$ can be used to keep c_g^i constant. Similar to the multiplier update (35), GRAMPC restricts the penalty to upper and lower bounds $c_{\min} \leq c_g^i \leq c_{\max}$, in order to avoid negligible values as well as unlimited growth of c_g^i. The lower penalty bound $c_{\min}$ is particularly relevant in case of MPC applications, where typically only a few iterations are performed in each MPC step. GRAMPC provides a support function that computes an estimate of $c_{\min}$ for the MPC problem at hand.

The updates (35) and (36) define the vector functions $\boldsymbol{\zeta_g}$, $\boldsymbol{\zeta_{g_T}}$ and $\boldsymbol{\xi_g}$, $\boldsymbol{\xi_{g_T}}$ in (17a) and (18a) with $N_{\boldsymbol{g}}$ and $N_{\boldsymbol{g}_T}$ components, corresponding to the number of equality and terminal equality constraints. Note that the multipliers for the equality and inequality constraints are time-dependent, i.e. $\boldsymbol{\mu_g}(t)$ and $\boldsymbol{\mu_h}(t)$, which implies that the functions $\boldsymbol{\zeta_g}$ and $\boldsymbol{\xi_g}$, resp. (35) and (36), are evaluated pointwise in time.

The inequality constrained case is handled in a similar spirit. For a single inequality constraint $\bar{h}^i := \bar{h}(\boldsymbol{x}^i, \boldsymbol{u}^i, \boldsymbol{p}^i, t, \mu_h, c_h)$, cf. (10a), the multiplier and penalty updates are defined by

$$\zeta_h(\mu_h^i, c_h^i, \bar{h}^i, \varepsilon_h) = \begin{cases} \mu_h^i + (1-\rho) c_h^i \bar{h}^i & \text{if } \left(\bar{h}^i > \varepsilon_h \wedge \eta^i \leq \varepsilon_{\text{rel,u}}\right) \vee \bar{h}^i < 0 \\ \mu_h^i & \text{else} \end{cases} \tag{37}$$

and

$$\xi_h(c_h^i, \bar{h}^i, \bar{h}^{i-1}, \varepsilon_h) = \begin{cases} \beta_{\text{in}}\, c_h^i & \text{if } \bar{h}^i \geq \max\left\{\gamma_{\text{in}} \bar{h}^{i-1}, \varepsilon_h\right\} \wedge \eta^i \leq \varepsilon_{\text{rel,u}} \\ \beta_{\text{de}}\, c_h^i & \text{else if } \bar{h}^i \leq \gamma_{\text{de}}\, \varepsilon_h \\ c_h^i & \text{else} \end{cases} \tag{38}$$

and constitute the vector functions $\boldsymbol{\zeta_h}$, $\boldsymbol{\zeta_{h_T}}$ and $\boldsymbol{\xi_h}$, $\boldsymbol{\xi_{h_T}}$ in (17b) and (18b) with $N_{\boldsymbol{h}}$ and $N_{\boldsymbol{h}_T}$ components. The condition $\bar{h}^i < 0$ in (37) ensures that the Lagrangian multiplier μ_h^i is reduced for inactive constraints, which corresponds to either $h^i < 0$ or $-\frac{\mu_h}{c_h}$ in view of the transformation (10).

4 Structure and usage of GRAMPC

This section describes the framework of GRAMPC and illustrates its general usage. GRAMPC is designed to be portable and executable on different operating systems and hardware without the use of external libraries. The code is implemented in plain C with a user-friendly interface to C++, Matlab/Simulink, and dSpace. The following lines give an overview of GRAMPC and demonstrate how to implement and solve a problem.

4.1 General structure

Figure 1 shows the general structure of GRAMPC and the steps that are necessary to compile an executable GRAMPC project. The first step in creating a new project is to define the problem using the provided C function templates, which will be detailed more in Sec 4.2. The user has the possibility to set problem specific parameters and algorithmic options concerning the numerical integrations in the gradient algorithm, the line search strategy as well as further preferences, also see Sec 4.3.

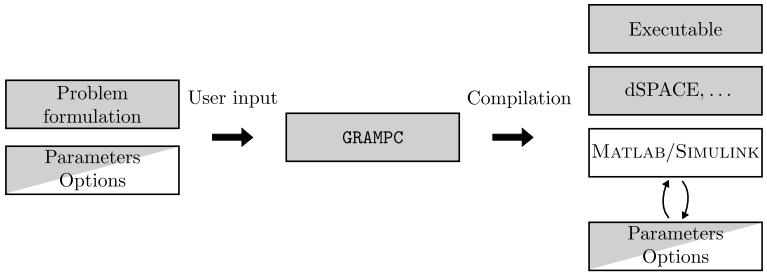

Fig. 1 General structure of GRAMPC (grey: C code, white: Matlab level)

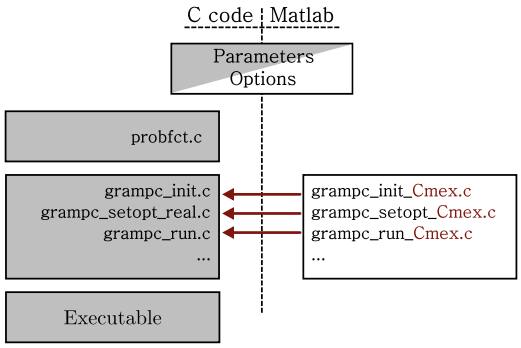

Fig. 2 Interfacing of GRAMPC to C (grey) and Matlab (white). Each GRAMPC function written in plain C has a corresponding Cmex function

A specific problem can be parameterized and numerically solved using C/C++, Matlab/Simulink, or dSpace. A closer look on this functionality is given in Fig. 2. The workspace of a GRAMPC project as well as algorithmic options and parameters are stored by the structure variable `grampc`. Several parameter settings are problem-specific and need to be provided, whereas other values are set to default values. A generic interface allows one to manipulate the `grampc` structure in order to set algorithmic options or parameters for the problem at hand. The functionalities of GRAMPC can be manipulated from Matlab/Simulink by means of mex routines that are wrappers for the corresponding C functions. This allows one to run a GRAMPC project with different parameters and options without recompilation.

4.2 Problem definition

The problem formulation in GRAMPC follows a generic structure. The essential steps for a problem definition are illustrated for the following MPC example

$$\min_{u(\cdot)} \quad J(u; \boldsymbol{x}_k) = \frac{1}{2}\Delta\boldsymbol{x}^\top(T)\boldsymbol{P}\Delta\boldsymbol{x}(T) + \frac{1}{2}\int_0^T \Delta\boldsymbol{x}^\top(T)\boldsymbol{Q}\Delta\boldsymbol{x} + R\Delta u^2 \,\mathrm{d}\tau \tag{39a}$$

$$\text{s.t.} \quad \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}-0.04\\ -7.01\end{bmatrix} u\,, \quad \begin{bmatrix}x_1(0)\\ x_2(0)\end{bmatrix} = \begin{bmatrix}x_{k,1}\\ x_{k,2}\end{bmatrix} \tag{39b}$$

$$\begin{bmatrix}-0.2\\ -0.1\end{bmatrix} \leq \begin{bmatrix}x_1\\ x_2\end{bmatrix} \leq \begin{bmatrix}0.01\\ 0.1\end{bmatrix}\,, \quad |u| \leq 0.0524 \tag{39c}$$

with $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{\text{des}}$, $\Delta u = u - u_{\text{des}}$, and the weights

$$\boldsymbol{P} = \boldsymbol{Q} = \begin{bmatrix}100 & 0\\ 0 & 10\end{bmatrix}\,, \quad R = 1\,. \tag{39d}$$

The dynamics (39b) are a simplified linear model of a single axis of a ball-on-plate system (Richter 2012) that is also included in the testbench of GRAMPC (see Sect. 5). The horizon length and the sampling time are set to $T = 0.3$ s and $\Delta t = 10$ms, respectively.

The problem formulation (39) is provided in the user template `probfct.c`. The following listing gives an expression of the function structure and the problem implementation for the ball-on-plate example (39).

```
/* Problem dimensions */
void ocp_dim(typeInt *Nx, typeInt *Nu, typeInt *Np, typeInt *Ng, typeInt
             *Nh, typeInt *NgT, typeInt *NhT, typeUSERPARAM *userparam)
{
  *Nx  = 2; *Nu  = 1;  *Np  = 0;
  *Nh  = 4; *Ng  = 0;
  *NgT = 0; *NhT = 0;
}

/* Right-hand side of dynamics M dx/dt = f(t,x,u,p,userparam) */
void ffct(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *u,
          ctypeRNum *p, typeUSERPARAM *userparam)
{
  out[0] = x[1]-0.04*u[0];
  out[1] =     -7.01*u[0];
}

/* Integral cost function int l(t,x(t),u(t),p,...) dt */
void lfct(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *u, ctypeRNum
          *p, ctypeRNum *xdes, ctypeRNum *udes, typeUSERPARAM *userparam)
{
  typeRNum* param = (typeRNum*)userparam;
  out[0] = 0.5*(param[0]*(x[0]-xdes[0])*(x[0]-xdes[0]) + \
                param[1]*(x[1]-xdes[1])*(x[1]-xdes[1]) + \
                param[2]*(u[0]-udes[0])*(u[0]-udes[0]) );
}

/* Terminal cost function V(T,x(T),p,xdes,userparam) */
void Vfct(typeRNum *out, ctypeRNum T, ctypeRNum *x, ctypeRNum *p,
          ctypeRNum *xdes, typeUSERPARAM *userparam)
{
  typeRNum* param = (typeRNum*)userparam;
  out[0] = 0.5*(param[3]*(x[0]-xdes[0])*(x[0]-xdes[0]) + \
                param[4]*(x[1]-xdes[1])*(x[1]-xdes[1]) );
}

/* Inequality constraints h(t,x(t),u(t),p,uperparam) <= 0 */
void hfct(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *u,
          ctypeRNum *p, typeUSERPARAM *userparam)
{
  typeRNum* param = (typeRNum*)userparam;
  out[0] =  param[5] - x[0];
  out[1] = -param[6] + x[0];
  out[2] =  param[7] - x[1];
  out[3] = -param[8] + x[1];
}
```

The naming of the functions follows the nomenclature of the general OCP formulation (1), except for the function `ocp_dim`, which defines the dimensions of the optimization problem. Problem specific parameters can be used inside the single functions via the pointer `userparam`. For convenience, the desired setpoint (`xdes`,`udes`) to be stabilized by the MPC is provided to the cost functions separately and therefore does not need to be passed via `userparam`.

In addition to the single OCP functions, GRAMPC requires the derivatives w.r.t. state $\boldsymbol{x}$, control $\boldsymbol{u}$, and if applicable w.r.t. the optimization parameters $\boldsymbol{p}$ and end time T, in order to evaluate the optimality conditions in Algorithm 2. Jacobians that occur in multiplied form, see e.g. $\left(\frac{\partial f}{\partial x}\right)^{\mathsf{T}} \boldsymbol{\lambda}$ in the adjoint dynamics (22), have to be provided in this representation. This helps to avoid unnecessary zero multiplications in case of sparse Jacobians. The following listing shows an excerpt of the corresponding gradient functions.

```
/* Multiplied Jacobian (df/dx)^T * mult */
void dfdx_mult(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *mult,
               ctypeRNum *u, ctypeRNum *p, typeUSERPARAM *userparam)
{
  out[0] = 0;
  out[1] = mult[0];
}
...
/* Jacobian dl/dx */
void dldx(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *u, ctypeRNum
          *p, ctypeRNum *xdes, ctypeRNum *udes, typeUSERPARAM *userparam)
{
  typeRNum* param = (typeRNum*)userparam;
  out[0] = param[0]*(x[0]-xdes[0]);
  out[1] = param[1]*(x[1]-xdes[1]);
}
...
/* Multiplied Jacobian (dh/dx)^T * mult */
void dhdx_mult(typeRNum *out, ctypeRNum t, ctypeRNum *x, ctypeRNum *u,
               ctypeRNum *p, ctypeRNum *mult, typeUSERPARAM *userparam)
{
  out[0] = -mult[0]+mult[1];
  out[1] = -mult[2]+mult[3];
}
...
```

4.3 Calling procedure and options

GRAMPC provides several key functions that are required for initializing and calling the MPC solver. As shown in Fig. 2, there exist mex wrapper functions that ensure that the interface for interacting with GRAMPC is largely the same under C/C++ and Matlab.

The following listing gives an idea on how to initialize GRAMPC and how to run a simple MPC loop for the ball-on-plate example under Matlab.

```
% user parameters
userparam = [100,10,1,100,10,-0.2,0.01,-0.1,0.1]

% initialization
grampc = grampc_init_Cmex(userparam);

% set parameters
grampc = grampc_setparam_Cmex(grampc,'x0',[0.1;0.01]);
grampc = grampc_setparam_Cmex(grampc,'xdes',[-0.2;0]);
grampc = grampc_setparam_Cmex(grampc,'u0',0);
grampc = grampc_setparam_Cmex(grampc,'udes',0);
grampc = grampc_setparam_Cmex(grampc,'Thor',0.3);
grampc = grampc_setparam_Cmex(grampc,'dt',0.01);
grampc = grampc_setparam_Cmex(grampc,'t0',0);
grampc = grampc_setparam_Cmex(grampc,'umin',0.0524);
grampc = grampc_setparam_Cmex(grampc,'umax',-0.0524);

% set options
grampc = grampc_setopt_Cmex(grampc,'Nhor',20);
grampc = grampc_setopt_Cmex(grampc,'MaxGradIter',2);
grampc = grampc_setopt_Cmex(grampc,'MaxMultIter',3);
grampc = grampc_setopt_Cmex(grampc,'InequalityConstraints','on');
grampc = grampc_setopt_Cmex(grampc,'Integrator','heun');

% MPC loop
for i = 1:iMPC
% run GRAMPC
grampc = grampc_run_Cmex(grampc);
...
% set new initial state
grampc = grampc_setparam_Cmex(grampc,'x0',grampc.sol.xnext);
...
end
...
```

The listing also shows some of the algorithmic settings, e.g. the number of discretization points `Nhor` for the horizon $[0, T]$, the maximum number of iterations $(i_{\max}, j_{\max})$ for Algorithm 1 and 2, or the choice of integration scheme for solving the canonical equations (22), (26). Currently implemented integration methods are (modified) Euler, Heun, 4th order Runge–Kutta as well as the solver RODAS (Hairer and Wanner 1996) that implements a 4th order Rosenbrock method for solving semi-implicit differential-algebraic equations with possibly singular and sparse mass matrix $\boldsymbol{M}$, cf. the problem definition in (1). The Euler and Heun methods use a fixed step size depending on the number of discretization points (`Nhor`), whereas RODAS and Runge–Kutta use adaptive step size control. The choice of integrator therefore has significant impact on the computation time and allows one to optimize the algorithm in terms of accuracy and computational efficiency. Further options not shown in the listing are e.g. the settings (`xScale`,`xOffset`) and (`uScale`,`uOffset`) in order to scale the input and state variables of the optimization problem.

The initialization and calling procedure for GRAMPC is largely the same under C/C++ and Matlab. One exception is the handling of user parameters in `userparam`. Under C, `userparam` can be an arbitrary structure, whereas the Matlab interface restricts `userparam` to be of type array (of arbitrary length). Moreover, the Matlab call of `grampc_run_Cmex` returns an updated copy of the `grampc` structure as output argument in order to comply with the Matlab policy to not manipulate input arguments.

5 Performance evaluation

The intention of this section is to evaluate the performance of GRAMPC under realistic conditions and for meaningful problem settings. To this end, an MPC testbench suite is defined to evaluate the computational performance in comparison to other state-of-the-art MPC solvers and to demonstrate the portability of GRAMPC to real-time and embedded hardware. The remainder of this section demonstrates the handling of typical problems from the field of MPC, moving horizon estimation and optimal control.

5.1 General MPC evaluation

The MPC performance of GRAMPC is evaluated for a testbench that covers a wide range of meaningful MPC applications. For the sake of comparison, the two MPC toolboxes ACADO and VIATOC are included in the evaluation, although it is not the intention of this section to rigorously rate the performance against other solvers, as such a comparison is difficult to carry out objectively. The evaluation should rather give a general impression about the performance and properties of GRAMPC. In addition, implementation details are presented for running the MPC testbench examples with GRAMPC on dSpace and ECU level.

5.1.1 Benchmarks

Table 1 gives an overview of the considered MPC benchmark problems in terms of the system dimension, the type of constraints (control/state/general nonlinear constraints), the dynamics (linear/nonlinear and explicit/semi-implicit) as well as the respective references. The MPC examples are evaluated with GRAMPC as well as with ACADO Toolkit (Houska et al. 2011) and VIATOC (Kalmari et al. 2015).

The testbench includes three linear problems (mass-spring-damper, helicopter, ball-on-plate) and one semi-implicit reactor example, where the mass matrix $\boldsymbol{M}$ in the semi-implicit form (1b) is computed from the original partial differential equation (PDE) using finite elements, also see Sect. 5.2.3. The nonlinear chain problem is a scalable MPC benchmark with m masses. Three further MPC examples are defined with nonlinear constraints. The permanent magnet synchronous machine (PMSM) possesses spherical voltage and current constraints in dq-coordinates, whereas the crane example with three degrees of freedom (DOF) and the vehicle problem include a nonlinear constraint to simulate a geometric restriction that must be bypassed (also see Sect. 5.2.1 for the crane example). Three of the problems (PMSM, 2D-crane, vehicle) are not evaluated with VIATOC, as nonlinear constraints cannot be handled by VIATOC at this stage.

For the GRAMPC implementation, most options are set to their default values. The only adapted parameters concern the horizon length T, the number of supporting points for the integration scheme and the integrator itself as well as the number of augmented Lagrangian and gradient iterations, $i_{\max}$ and $j_{\max}$, respectively. Default settings are used for the multiplier and penalty updates for the sake of consistency,

Table 1 Overview of MPC benchmark problems

Problem	Dimensions		Constraints			Dynamics		References
	N_x	N_u	$\boldsymbol{u}$	$\boldsymbol{x}$	nonl.	semi-impl.	Linear	
Mass-spring-damper	10	2	Yes	No	No	No	Yes	Käpernick (2016)
Motor (PMSM)	4	2	Yes	Yes	Yes	No	No	Englert and Graichen (2018)
Nonl. chain ($m = 4$)	21	3	Yes	No	No	No	No	Kirches et al. (2012)
Nonl. chain ($m = 6$)	33	3	Yes	No	No	No	No	Kirches et al. (2012)
Nonl. chain ($m = 8$)	45	3	Yes	No	No	No	No	Kirches et al. (2012)
Nonl. chain ($m = 10$)	57	3	Yes	No	No	No	No	Kirches et al. (2012)
2D-Crane	6	2	Yes	Yes	Yes	No	No	Käpernick and Graichen (2013)
3D-Crane	10	3	Yes	No	No	No	No	Graichen et al. (2010)
Helicopter	6	2	Yes	Yes	No	No	Yes	Tøndel and Johansen (2002)
Quadrotor	9	4	Yes	No	No	No	No	Käpernick and Graichen (2014a)
VTOL	6	2	Yes	No	No	No	No	Sastry (2013)
Ball-on-plate	2	1	Yes	Yes	No	No	Yes	Richter (2012)
Vehicle	5	2	Yes	No	Yes	No	No	Werling et al. (2012)
CSTR reactor	4	2	Yes	No	No	No	No	Rothfuss et al. (1996)
PDE reactor	11	1	Yes	No	No	Yes	No	Utz et al. (2010)

see Algorithm 1 as well as Sect. 3.5. Note, however, that the performance and computation time of GRAMPC can be further optimized by tuning the parameters related to the penalty update to a specific problem. All benchmark problems listed in Table 1 are available as example implementations in GRAMPC.

ACADO and VIATOC are individually tuned for each MPC problem by varying the number of shooting intervals and iterations in order to either achieve minimum computation time (setting "speed") or optimal performance in terms of minimal cost at reasonable computational load (setting "optimal"). The solution of the quadratic programs of ACADO was done with qpOASES (Ferreau et al. 2014).

The single MPC projects are integrated in a closed-loop simulation environment with a fourth-order Runge–Kutta integrator with adaptive step size to ensure an accurate system integration regardless of the integration schemes used internally by the MPC toolboxes. The MPC is initialized with the stationary solution at the start of the simulation scenario. The evaluation was carried out on a Windows 10 machine with Intel(R) Core(TM) i5-5300U CPU running at 2.3GHz using the Microsoft Visual C++ 2013 Professional (C) compiler. Each simulation was run multiple times to obtain a reliable average computation time.

5.1.2 Evaluation results

Table 2 shows the evaluation results for the benchmark problems in terms of computation time and cost value integrated over the whole time interval of the simulation scenario. The cost values are normalized to the best one of each benchmark problem. The results for ACADO and VIATOC are listed for the settings "speed" and "optimal", as mentioned above. The depicted computation times are the mean computation times, averaged over the complete time horizon of the simulation scenario. The best values for computation time and performance (minimal cost) for each benchmark problem are highlighted in bold font.

The linear MPC problems (mass-spring-damper, helicopter, ball-on-plate) with quadratic cost functions can be tackled well by VIATOC and ACADO due to their underlying linearization techniques. The PDE reactor problem contains a stiff system dynamics in semi-implicit form. ACADO can handle such problems well using its efficient integration schemes, whereas VIATOC relies on fixed step size integrators and therefore requires a relatively large amount of discretization points. While GRAMPC achieves the fastest computation time, the cost value of both ACADO settings as well as the VIATOC optimal setting is lower. A similar cost function, however, can be achieved by GRAMPC when deviating from the default parameters.

In case of the state constrained 2D-crane problem, the computation time is higher for ACADO than for GRAMPC. This appears to be due to the fact that almost over the complete simulation time a nonlinear constraint of a geometric object to be bypassed is active. A closer look at this problem is taken in Sect. 5.2.1.

The CSTR reactor example possesses state and control variables in different orders of magnitude and therefore benefits from scaling. Since GRAMPC supports scaling natively, the computation time is faster than for VIATOC, where the scaling would have to be done manually. Due to the Hessian approximation used by ACADO, it is far less affected by the different scales in the states and controls. Figure 3 shows the state and control trajectories of the three different solvers for the CSTR example. As indicated by the values of the cost function in Table 2, the computed trajectories are quite similar. Only for the first hour, small differences can be observed for the speed settings of ACADO and VIATOC. This can also be seen in Fig. 4, where the cost function value in each MPC step is shown. Slightly different values can be observed, but after about 0.4 h all solvers reach the same stationary value.

A large difference in the cost values occurs for the VTOL example (Vertical Take-Off and Landing Aircraft). Due to the nonlinear dynamics and the corresponding coupling of the control variables, it seems that the gradient method underlying the minimization steps of GRAMPC is more accurate and robust when starting in an equilibrium position than the iterative linearization steps undertaken by ACADO and VIATOC.

The scaling behavior of the MPC schemes w.r.t. the problem dimension is investigated for the nonlinear chain in Table 2. Four different numbers of masses are considered, corresponding to 21–57 state variables and three controls. Although the algorithmic complexity of the augmented Lagrangian/gradient projection algorithm of GRAMPC grows linearly with the state dimension, this is not exactly the case for the nonlinear chain, as the stiffness of the dynamics increases for a larger number of masses, which leads to smaller step sizes of the adaptive Runge–Kutta integrator

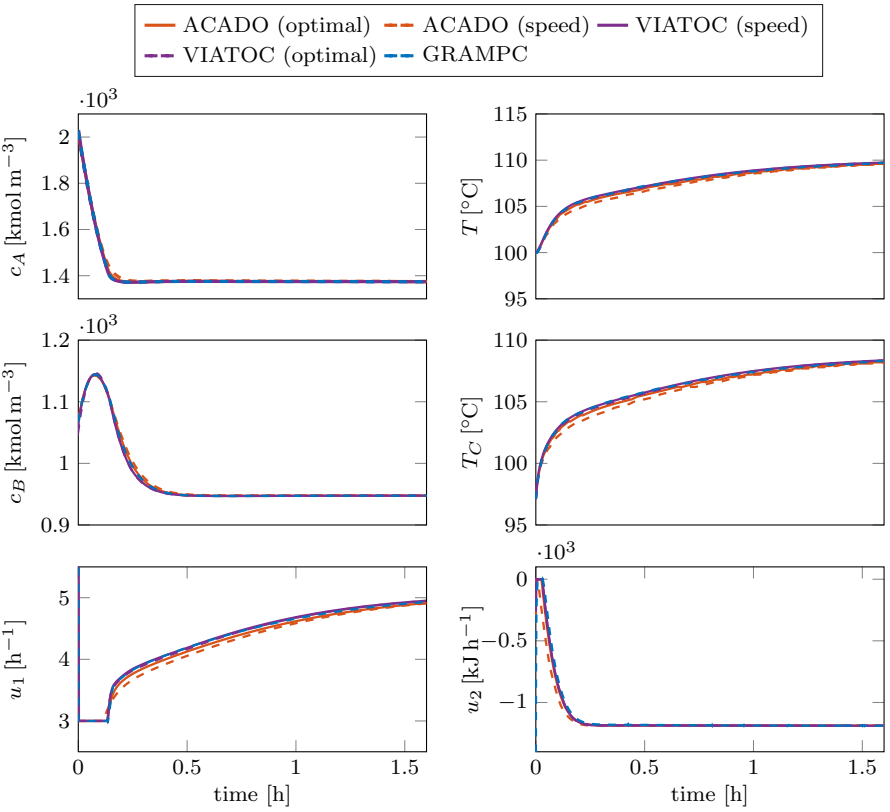

Fig. 3 MPC trajectories for the CSTR reactor using the different solvers

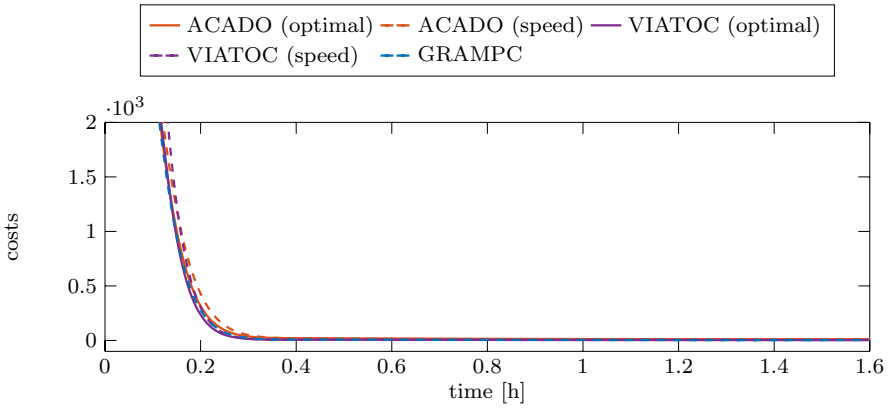

Fig. 4 MPC cost trajectory for the CSTR reactor using the different solvers

Table 2 Evaluation results for the benchmark problems in Table 1 with overall integrated cost J_{int} and computation time t_{CPU} in milliseconds

Problem	GRAMPC		ACADO (optimal)		ACADO (speed)		VIATOC (optimal)		VIATOC (speed)	
	J_{int}	t_{CPU}	J_{int}	t_{CPU}	J_{int}	t_{CPU}	J_{int}	t_{CPU}	J_{int}	t_{CPU}
Mass-spring-damper	**1.000**	0.060	1.030	0.370	1.351	0.110	**1.000**	0.075	1.221	**0.049**
Motor (PMSM)	1.006	**0.032**	**1.000**	0.129	1.096	0.057	–	–	–	–
Nonl. chain ($m = 4$)	1.003	**0.301**	1.003	4.666	1.027	1.770	**1.000**	12.32	1.041	7.660
Nonl. chain ($m = 6$)	**1.000**	**0.707**	**1.000**	12.438	1.028	5.407	**1.000**	18.10	1.042	11.48
Nonl. chain ($m = 8$)	**1.000**	**1.158**	1.005	26.834	1.050	10.127	**1.000**	38.51	1.055	15.30
Nonl. chain ($m = 10$)	**1.000**	**1.463**	1.004	43.467	1.042	21.745	1.004	52.84	1.149	24.85
2D-Crane	1.032	**0.019**	**1.000**	0.446	1.096	0.058	–	–	–	–
3D-Crane	1.002	**0.036**	**1.000**	0.728	1.013	0.321	1.163	0.839	1.160	0.194
Helicopter	1.017	0.054	1.006	0.163	1.096	**0.045**	**1.000**	0.185	1.060	0.071
Quadrotor	**1.000**	**0.022**	**1.000**	1.465	1.005	0.243	1.009	0.535	1.010	0.113
VTOL	**1.000**	**0.033**	1.210	0.229	1.227	0.090	1.243	0.092	1.259	0.083
Ball-on-plate	1.113	**0.014**	1.112	0.068	1.126	0.022	**1.000**	0.116	1.158	0.018
Vehicle	1.027	**0.092**	**1.000**	1.313	1.003	0.322	–	–	–	–
CSTR reactor	**1.000**	**0.058**	**1.000**	0.195	1.002	0.076	1.001	0.365	1.017	0.140
PDE reactor	1.059	**0.362**	**1.000**	6.259	1.005	0.498	1.046	7.214	1.072	2.868

that was used in GRAMPC for this problem. ACADO shows a more significant increase in computation time for larger values of m, which was to be expected in view of the SQP algorithm underlying ACADO. The computation time for VIATOC is worse for this example, since only fixed step size integrators are available in the current release, which requires to increase the number of discretization points manually. Figure 5 shows a logarithmic plot of the computation time for all three MPC solvers plotted over the number of masses of the nonlinear chain.

The computation times shown in Table 2 are average values and therefore give no direct insight into the real-time feasibility of the MPC solvers and the variation of the computational load over the single sampling steps. To this end, Fig. 6 shows accumulation plots of the computation time per MPC step for three selected problems of the testbench. The computation times were evaluated after 30 successive runs to obtain reliable results. The plots show that the computation time of GRAMPC is almost constant for each MPC iteration, which is important for embedded control applications and to give tight upper bounds on the computation time for

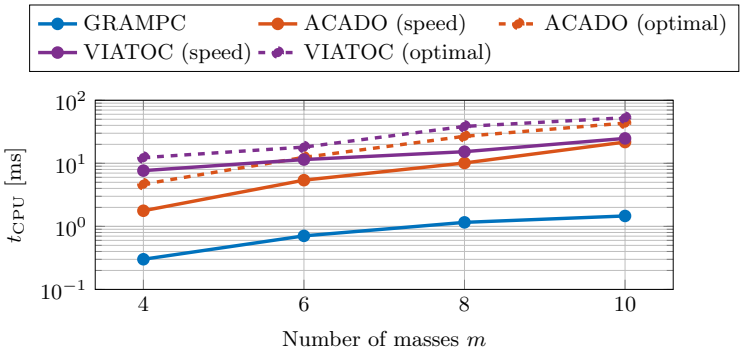

Fig. 5 Computation time for the nonlinear chain example (also see Table 2)

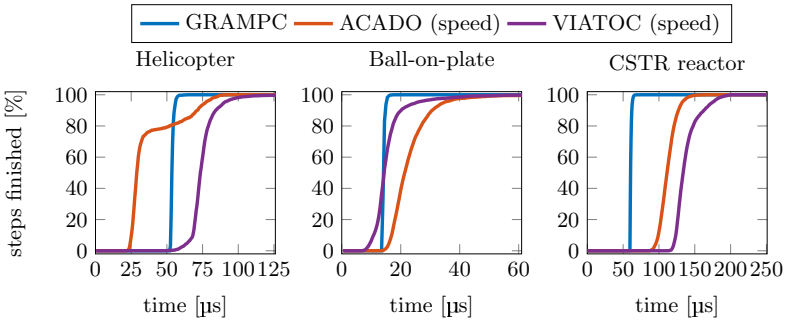

Fig. 6 Number of iterations that finish below a certain computation times for three different examples (averaged over 30 runs)

real-time guarantees. ACADO and VIATOC show a larger variation of the computation time over the iterations, which is mainly due to the active set strategy that both solvers follow and the varying number of QP iterations in each real-time iteration of ACADO, c.f. Houska et al. (2011).

In conclusion, it can be said that GRAMPC has overall fast and real-time feasible computation times for all benchmark problems, in particular for nonlinear systems and in connection with (nonlinear) constraints. Furthermore, GRAMPC scales well with increasing system dimension.

5.1.3 Embedded realization

In addition to the general MPC evaluation, this section evaluates the computation time and memory requirements of GRAMPC for the benchmark problems on real-time and embedded hardware. GRAMPC was implemented on a dSpace MicroLabbox (DS1202) with a 2 GHz Freescale QolQ processor as well as on the microntroller RH850/P1M from Renesas with a CPU frequency of 160 MHz, 2 MB program flash and 128 kB RAM. This processor is typically used in electronic control units (ECU) in the automotive industry. The GRAMPC implementation on this microcontroller therefore can be seen as a prototypical ECU realization. As it is commonly done in embedded systems, GRAMPC was implemented using single floating point precision on both systems due to the floating point units of the processors.

Table 3 lists the computation time and RAM memory footprint of GRAMPC on both hardware platforms for the testbench problems in Tables 1 and 2. The settings of GRAMPC are the same as in the previous section, except for the floating point precision. Due to the compilation size limit of the ECU compiler (< 10 kB), the nonlinear chain examples as well as the PDE reactor could not be compiled on the ECU.

The computation times on the dSpace hardware are below the sampling time for all example problems. The same holds for the ECU implementation, except for the 2D-crane, the PMSM, and the VTOL example. However, as mentioned before, tuning of the algorithm can further reduce the runtime, as most of the multiplier and penalty update parameters are taken as default. Note that there is no constant scaling factor between the computation times on dSpace and ECU level, which is probably due to the different realization of the math functions by the respective floating point unit/compiler[1] on the different hardware.

The required memory is below 9 kB for all examples, except for the nonlinear chain and the PDE reactor, which is less than 7% of the available RAM on the considered ECU. Although the nonlinear chain and the PDE reactor could not be compiled on the ECU as mentioned above, the memory usage as well as the computation time increase only linearly with the size of the system (using the same GRAMPC settings). Overall, the computational speed and the small memory footprint demonstrate the applicability of GRAMPC for embedded systems.

[1] For example software or hardware realization of sine or cosine functions.

Table 3 Computation time and memory usage for the embedded realization of GRAMPC on dSpace hardware (DS1202) and ECU level (Renesas RH850/P1M) with single floating point precision

Problem	t_{DSpace} (ms)	t_{ECU} (ms)	Memory (kB)
Mass-spring-damper	0.24	4.00	6.5
Motor (PMSM)	0.13	2.10	2.4
Nonl. chain ($m = 4$)	4.79	–	12.9
Nonl. chain ($m = 6$)	9.62	–	18.5
Nonl. chain ($m = 8$)	17.50	–	24.1
Nonl. chain ($m = 10$)	24.20	–	29.8
2D-Crane	0.18	1.65	4.5
3D-Crane	0.31	3.05	7.3
Helicopter	0.11	1.88	5.8
Quadrotor	0.21	1.60	4.4
VTOL	0.37	2.80	6.2
Ball-on-plate	0.05	0.92	2.0
Vehicle	0.25	2.69	3.6
CSTR reactor	0.43	6.81	5.1
PDE reactor	6.48	–	15.0

5.2 Application examples

This section discusses some application examples, including a more detailed view on two selected problems from the testbench collection (state constrained and semi-implicit problem), a shrinking horizon MPC application, an equality constrained OCP as well as a moving horizon estimation problem.

5.2.1 Nonlinear constrained model predictive control

The 2D-crane example in Tables 1 and 2 is a particularly challenging one, as it is subject to a nonlinear constraint that models the collision avoidance of an object or obstacle. A schematic representation of the overhead crane is given in Fig. 7. The crane has three degrees-of-freedom and the nonlinear dynamics read as (Käpernick and Graichen 2013)

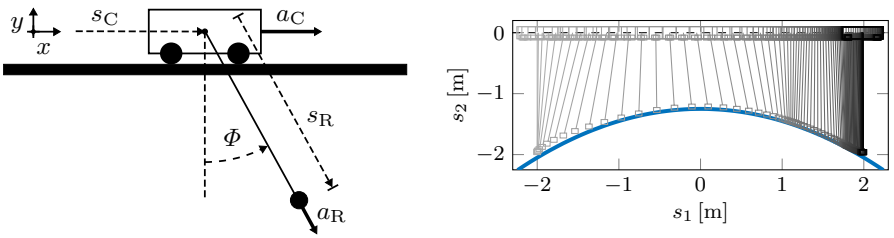

Fig. 7 Schematic representation of the overhead crane (left) and simulated crane movement from the initial state to the desired setpoint (right)

$$\ddot{s}_\mathrm{C} = u_1, \quad \ddot{s}_\mathrm{R} = u_2, \quad \ddot{\phi} = -\frac{1}{s_\mathrm{R}}\left(g\sin(\phi) + \ddot{s}_\mathrm{C}\cos(\phi) + 2\dot{s}_\mathrm{R}\dot{\phi}\right)$$

with the gravitational constant g. The system state $\boldsymbol{x} = [s_\mathrm{C}, \dot{s}_\mathrm{C}, s_\mathrm{R}, \dot{s}_\mathrm{R}, \phi, \dot{\phi}]^\mathsf{T} \in \mathbb{R}^6$ comprises the cart position s_C, the rope length s_R, the angular deflection ϕ and the corresponding velocities. The two controls $\boldsymbol{u} \in \mathbb{R}^2$ are the cart acceleration u_1 and the rope acceleration u_2, respectively.

The cost functional (2b) consists of the integral part

$$l(\boldsymbol{x}, \boldsymbol{u}) = (\boldsymbol{x} - \boldsymbol{x}_\mathrm{des})^\mathsf{T}\boldsymbol{Q}(\boldsymbol{x} - \boldsymbol{x}_\mathrm{des}) + (\boldsymbol{u} - \boldsymbol{u}_\mathrm{des})^\mathsf{T}\boldsymbol{R}(\boldsymbol{u} - \boldsymbol{u}_\mathrm{des}), \tag{40}$$

which penalizes the deviation from the desired setpoints $\boldsymbol{x}_\mathrm{des} \in \mathbb{R}^6$ and $\boldsymbol{u}_\mathrm{des} \in \mathbb{R}^2$ respectively. The weight matrices are set to $\boldsymbol{Q} = \mathrm{diag}(1, 2, 2, 1, 1, 4)$ and $\boldsymbol{R} = \mathrm{diag}(0.05, 0.05)$ (omitting units). The controls and angular velocity are subject to the box constraints $|u_i| \leq 2\,\mathrm{m\,s^{-2}}$, $i \in 1, 2$ and $|\dot{\phi}| \leq 0.3\,\mathrm{rad\,s^{-1}}$, $i \in 1, 2$. In addition, the nonlinear inequality constraint

$$h(\boldsymbol{x}) = \cos(\phi)s_\mathrm{R} - 0.2\,\mathrm{m}^{-1}\left(s_\mathrm{C} + \sin(\phi)s_\mathrm{R}\right)^2 + 1.25\,\mathrm{m} \leq 0 \tag{41}$$

is imposed, which represents a geometric security constraint, e.g. for trucks, over which the load has to be lifted, see Fig. 8 (right).

The prediction horizon and sampling time for the crane problem are set to $T = 2\,\mathrm{s}$ and $\Delta t = 2\,\mathrm{ms}$, respectively. The number of augmented Lagrangian steps and inner gradient iterations of GRAMPC are set to $(i_\mathrm{max}, j_\mathrm{max}) = (1, 2)$. These settings correspond to the computational results in Tables 2 and 3.

The right part of Fig. 7 illustrates the movement of the overhead crane from the initial state $\boldsymbol{x}_0 = [-2\,\mathrm{m}, 0, 2\,\mathrm{m}, 0, 0, 0]^\mathsf{T}$ to the desired setpoint $\boldsymbol{x}_\mathrm{des} = [2\,\mathrm{m}, 0, 2\,\mathrm{m}, 0, 0, 0]^\mathsf{T}$. Figure 8 shows the corresponding trajectories of the states $\boldsymbol{x}(t)$ and controls $\boldsymbol{u}(t)$ as well as the nonlinear constraint (41) plotted as time function $h(\boldsymbol{x}(t))$. This transition problem is quite challenging, since the nonlinear constraint (41) is active for more than half of the simulation time. One can slightly see an initial violation of the constraint $h(\boldsymbol{x})$, which should be negligible in practical applications. Nevertheless, one can satisfy the constraint to a higher accuracy by increasing the number of iterations $(i_\mathrm{max}, j_\mathrm{max})$, in particular of the augmented Lagrangian iterations. This behaviour is pictured in Fig. 9, where the cost function as well as the necessary computation time are shown with regard to the corresponding iteration count. It can be observed that the gradient algorithm quickly converges to a good solution and with increasing computation time (meaning increased iteration count, cf. the middle plot) the optimality of the solution improves further. The right plot of Fig. 9 shows the relative deviation from the optimal cost, i.e. $(J - J_\mathrm{opt})/J_\mathrm{opt}$, in relation to the necessary computation time. For each computation time there exists an optimal iteration combination, shown by the Pareto front (dashed black line) in the right plot of Fig. 9.

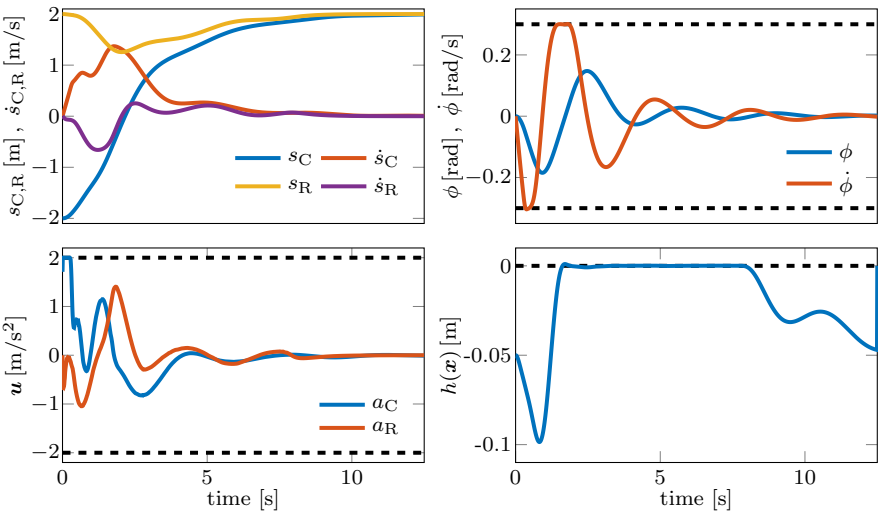

Fig. 8 MPC trajectories for the 3DOF crane

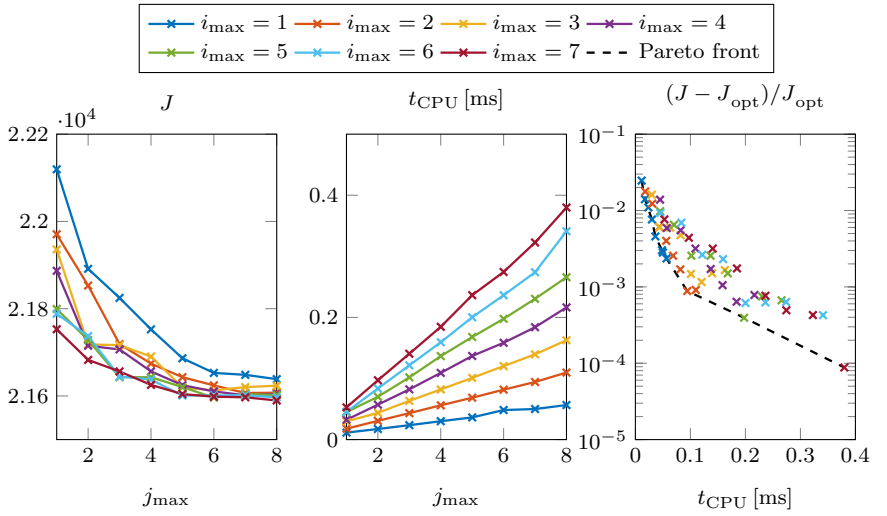

Fig. 9 MPC cost (left) and computation time (middle) for different iteration combinations. The right plot shows the relative deviation from the optimal cost w.r.t. the necessary computation time, as well as the corresponding Pareto front (dashed line)

5.2.2 MPC on shrinking horizon

"Classical" MPC with a constant horizon length typically acts as an asymptotic controller in the sense that a desired setpoint is only reached asymptotically. MPC on a shrinking horizon instead reduces the horizon time T in each sampling step in order to reach the desired setpoint in finite time. In particular, if the desired setpoint is

incorporated into a terminal constraint and the prediction horizon T is optimized in each MPC step, then T will be automatically reduced over the runtime due to the principle of optimality.

Shrinking horizon MPC with GRAMPC is illustrated for the following double integrator problem

$$\min_{u,T} \quad J(u,T;\boldsymbol{x}_k) = T + \frac{1}{2}\int_0^T r u^2(\tau)\,\mathrm{d}\tau \tag{42a}$$

$$\text{s.t.} \quad \dot{x}_1(\tau) = x_2(\tau)\,, \quad x_1(0) = x_{1,k} = x_1(t_k) \tag{42b}$$

$$\dot{x}_2(\tau) = u(\tau)\,, \quad x_2(0) = x_{2,k} = x_2(t_k) \tag{42c}$$

$$|u(\tau)| \leq 1 \quad \tau \in [0,T] \tag{42d}$$

$$\boldsymbol{x}(T) = \boldsymbol{x}_{\text{des}} \tag{42e}$$

with the state $\boldsymbol{x} = [x_1, x_2]^\mathsf{T}$ and control u subject to the box constraint (42d). The weight $r > 0$ in the cost functional (42a) allows a trade-off between energy optimality and time optimality of the MPC. The desired setpoint $\boldsymbol{x}_{\text{des}}$ is added as terminal constraint (42e), i.e. $\boldsymbol{g}_T(\boldsymbol{x}(T)) := \boldsymbol{x}(T) - \boldsymbol{x}_{\text{des}} = \boldsymbol{0}$ in view of (1c), and the prediction horizon T is treated as optimization variable in addition to the control $u(\tau)$, $\tau \in [0,T]$.

For the simulations, the weight in the cost functional is set to $r = 0.01$ and the initial value of the horizon length is chosen as $T = 6$. The iteration numbers for GRAMPC are set to $(i_{\max}, j_{\max}) = (1,2)$ in conjunction with a sampling time of $\Delta t = 0.001$ in order to resemble a real-time implementation. Figure 10 shows the simulation results for the double integrator problem with the desired setpoint

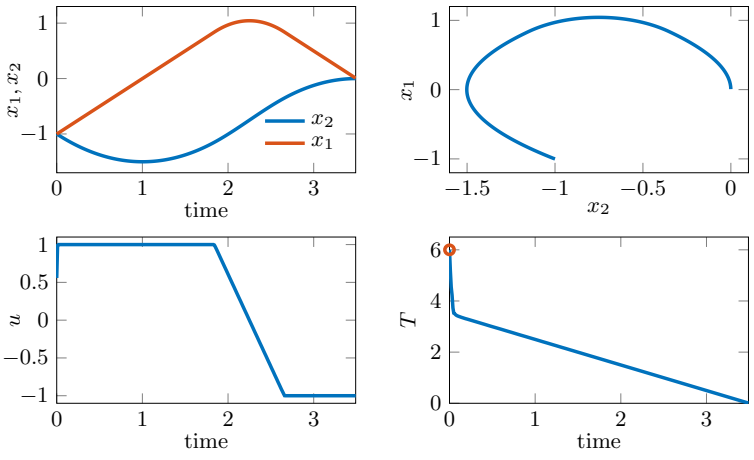

Fig. 10 MPC trajectories with shrinking horizon for the double integrator problem (42)

$\boldsymbol{x}_{\text{des}} = \mathbf{0}$ and the initial state $\boldsymbol{x}(0) = [-1, -1]^{\mathrm{T}}$. Obviously, the state trajectories reach the origin in finite time corresponding to the anticipated behavior of the shrinking horizon MPC scheme. The lower right plot of Fig. 10 shows the temporal evolution of the horizon length T over the runtime t. The initial end time of $T = 6$ is marked as a red circle. In the first MPC steps, the optimization quickly decreases the end time to approximately $T = 3.5$. In this short time interval, GRAMPC produces a suboptimal solution due to the strict limitation of the iterations $(i_{\max}, j_{\max})$. Afterwards, however, the prediction horizon declines linearly, which concurs with the principle of optimality and shows the optimality of the computed trajectories after the initialization phase. In the simulated example, this knowledge is incorporated in the MPC implementation by substracting the sampling time Δt from the last solution of T for warm-starting the next MPC step. The simulation is stopped as soon as the horizon length reaches its minimum value $T_{\min} = \Delta t = 0.1$.

5.2.3 Semi-implicit problems

The system formulations that can be handled with GRAMPC include DAE systems with singular mass matrix $\boldsymbol{M}$ as well as general semi-implicit systems. An application example is the discretization of spatially distributed systems by means of finite elements. This is illustrated for a quasi-linear diffusion–convection–reaction system, which is also implemented in the testbench (PDE reactor example). The thermal behavior of the reactor is described on the one-dimensional spatial domain $z = (0, 1)$ using the PDE formulation (Utz et al. 2010)

$$\partial_t \theta = \partial_z \left[(q_0 + q_1 \theta) \partial_z \theta - \nu \theta \right] + (r_0 + r_1 \theta) \theta \tag{43a}$$

with the boundary and initial conditions

$$\partial_z \theta|_{z=0} = 0 \tag{43b}$$

$$\partial_z \theta|_{z=1} + \theta(1, t) = u \tag{43c}$$

$$\theta(\cdot, 0) = \theta_0 \tag{43d}$$

for the temperature $\theta = \theta(z, t)$. The process is controlled by the boundary control $u = u(t)$. Diffusive and convective processes of the reactor are modeled by the nonlinear heat equation (43a) with the parameters $q_1 = 2$, $q_2 = -0.05$, and $\nu = 1$, respectively. Reaction effects are included using the parameters $r_0 = 1$ and $r_1 = 0.2$. The Neumann boundary condition (43b), the Robin boundary condition (43c), and the initial condition (43) complete the mathematical description of the system dynamics. Both spatial and time domain are normalized for the sake of simplicity. A more detailed description of the system dynamics can be found in Utz et al. (2010).

The PDE system (43) is approximated by an ODE system of the form (16a) by applying a finite element discretization technique (Zienkiewicz and Morgan 1983), whereby the new state variables $\boldsymbol{x} \in \mathbb{R}^{N_x}$ approximate the temperature θ on the discretized spatial domain z with $N_{\boldsymbol{x}}$ spatial grid points. The finite element discretization eventually leads to a finite-dimensional system dynamics of the form $\boldsymbol{M}\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u)$ with the mass matrix $\boldsymbol{M} \in \mathbb{R}^{N_x \times N_x}$ and the nonlinear system function

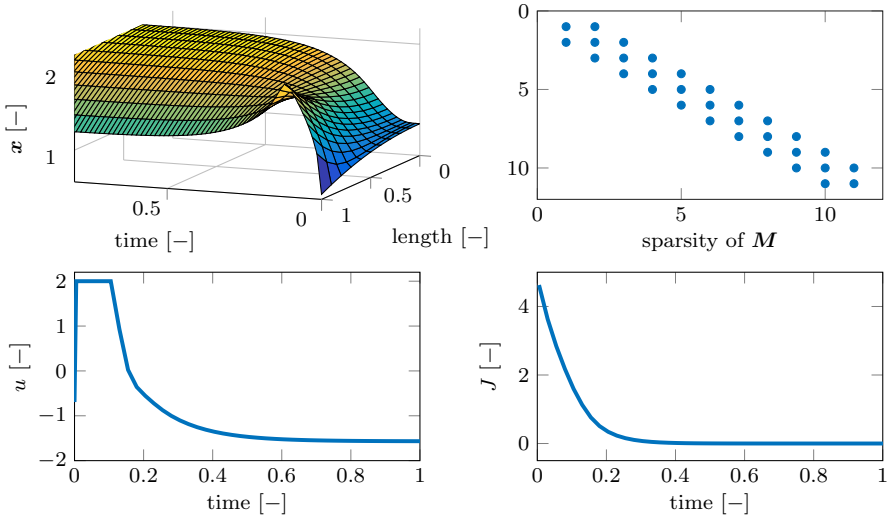

Fig. 11 Simulated MPC trajectories for the PDE reactor (33)

$\boldsymbol{f}(\boldsymbol{x}, u)$. In particular, $N_{\boldsymbol{x}} = 11$ grid points are chosen for the GRAMPC simulation of the reactor (43). The upper right plot of Fig. 11 shows the sparsity structure of the mass matrix $\boldsymbol{M} \in \mathbb{R}^{11\times 11}$.

The control task for the reactor is the stabilization of a stationary profile $\boldsymbol{x}_{\text{des}}$ which is accounted for in the quadratic MPC cost functional

$$J(u; \boldsymbol{x}_k) := \tfrac{1}{2}\left\|\boldsymbol{x}(T) - \boldsymbol{x}_{\text{des}}\right\|^2 + \int_0^T \tfrac{1}{2}\left\|\boldsymbol{x}(t) - \boldsymbol{x}_{\text{des}}\right\|^2 + 10^{-2}\left\|u - u_{\text{des}}\right\|^2 \mathrm{d}t\,.$$

The desired setpoint $(\boldsymbol{x}_{\text{des}}, u_{\text{des}})$ as well as the initial values $(\boldsymbol{x}_0, u_0)$ are numerically determined from the stationary differential equation

$$0 = \partial_z\left[(q_0 + q_1\theta(z,\tau))\partial_z\theta(z,\tau) - \nu\theta(z,\tau)\right] + (r_0 + r_1\theta(z,\tau))\theta(z,\tau) \tag{44}$$

with the corresponding boundary conditions

$$\theta(0,0) = 1, \quad \partial_z\theta(0,0) = 0$$
$$\theta(0,\infty) = 2, \quad \partial_z\theta(0,\infty) = 0$$

The prediction horizon and sampling time of the MPC scheme are set to $T = 0.2$ and $\Delta t = 0.005$. The number of iterations are limited by $(i_{\max}, j_{\max}) = (1, 2)$. The box constraints for the control are chosen as $|u(t)| \leq 2$.

The numerical integration in GRAMPC is carried out using the solver RODAS (Hairer and Wanner 1996). The sparse numerics of RODAS allow one to cope with the banded structure of the matrices in a computationally efficient manner. Figure 11 shows the setpoint transition from the initial temperature profile $\boldsymbol{x}_0$ to the desired

temperature $\boldsymbol{x}_{\text{des}} = [2.00, 1.99, 1.97, 1.93, 1.88, 1.81, 1.73, 1.63, 1.51, 1.38, 1.23]^\mathsf{T}$ and desired control $u_{\text{des}} = -1.57$.

5.2.4 OCP with equality constraints

An illustrative example of an optimal control problem with equality constraints is a dual arm robot with closed kinematics, e.g. to handle or carry work pieces with both arms. For simplicity, a planar dual arm robot with the joint angles (x_1, x_2, x_3) and (x_4, x_5, x_6) for the left and right arm is considered. The dynamics are given by simple integrators

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x}_1 & \dot{x}_2 & \dot{x}_3 & \dot{x}_4 & \dot{x}_5 & \dot{x}_6 \end{bmatrix}^\mathsf{T} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{bmatrix}^\mathsf{T} = \boldsymbol{u} \tag{45}$$

with the joint velocities as control input $\boldsymbol{u}$. Given the link lengths $\boldsymbol{a} = [a_1, a_2, a_3, a_4, a_5, a_6]$, the forward kinematics of left and right arm are computed by

$$\boldsymbol{p}_{\text{L}}(\boldsymbol{x}) = \begin{bmatrix} a_1 \cos(x_1) + a_2 \cos(x_1 + x_2) + a_3 \cos(x_1 + x_2 + x_3) \\ a_1 \sin(x_1) + a_2 \sin(x_1 + x_2) + a_3 \sin(x_1 + x_2 + x_3) \\ x_1 + x_2 + x_3 \end{bmatrix} \tag{46}$$

and

$$\boldsymbol{p}_{\text{R}}(\boldsymbol{x}) = \begin{bmatrix} 1 + a_4 \cos(x_4) + a_5 \cos(x_4 + x_5) + a_6 \cos(x_4 + x_5 + x_6) \\ a_4 \sin(x_4) + a_5 \sin(x_4 + x_5) + a_6 \sin(x_4 + x_5 + x_6) \\ x_4 + x_5 + x_6 \end{bmatrix}. \tag{47}$$

The closed kinematic chain is enforced by the equality constraint

$$\boldsymbol{g}(\boldsymbol{x}) := \boldsymbol{p}_{\text{L}}(\boldsymbol{x}) - \boldsymbol{p}_{\text{R}}(\boldsymbol{x}) - [0,\ 0,\ \pi]^\mathsf{T} = \boldsymbol{0}\,. \tag{48}$$

A point-to-point motion from $\boldsymbol{x}_0 = [\frac{\pi}{2}, -\frac{\pi}{2}, 0, -\frac{\pi}{2}, \frac{\pi}{2}, 0]^\mathsf{T}$ to $\boldsymbol{x}_f = [-\frac{\pi}{2}, \frac{\pi}{2}, 0, \frac{\pi}{2}, -\frac{\pi}{2}, 0]^\mathsf{T}$ is considered as control task, which is formulated as the optimal control problem

$$\min_{\boldsymbol{u}} \ J(\boldsymbol{u}) := \int_0^T \frac{1}{2} \boldsymbol{u}(t)^\mathsf{T} \boldsymbol{R} \boldsymbol{u}(t)\, \mathrm{d}t \tag{49a}$$

$$\text{s.t.} \quad \dot{\boldsymbol{x}}(t) = \boldsymbol{u}(t),\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \quad \boldsymbol{x}(T) = \boldsymbol{x}_f \tag{49b}$$

$$\boldsymbol{g}(\boldsymbol{x}(t)) = \boldsymbol{0}, \boldsymbol{u}(t) \in \left[\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}\right] \tag{49c}$$

with the fixed end time $T = 10\,\text{s}$ and the control constraints $-\boldsymbol{u}_{\min} = \boldsymbol{u}_{\max} = [1, 1, 1, 1, 1, 1]^\mathsf{T}\text{s}^{-1}$, which limit the angular speeds of the robot arms. The cost functional minimizes the squared joint velocities with $\boldsymbol{R} = \boldsymbol{I}_6$.

Table 4 shows the computation results of GRAMPC for solving OCP (49a) with increasingly restrictive values of the gradient tolerance $\varepsilon_{\text{rel,c}}$ and constraint tolerance $\boldsymbol{\varepsilon_g}$ that are used for checking convergence of Algorithm 1 and 2. The required

Table 4 Computation results for the planar two-arm robot with closed kinematics

Gradient tot. $\varepsilon_{\text{rel,c}}$	Constraint tot. $\boldsymbol{\varepsilon_g}$	Gradient iterations i (avg.)	Augm. Lagr. iterations j	t_{CPU} (ms)
1×10^{-5}	1×10^{-3}	64	189	135
1×10^{-6}	1×10^{-4}	65	298	214
1×10^{-7}	1×10^{-5}	306	190	628
1×10^{-8}	1×10^{-6}	222	431	1015

computation time t_{CPU} as well as the average number of (inner) gradient iterations and the number of (outer) augmented Lagrangian iterations are shown in Table 4. The successive reduction of the tolerances $\varepsilon_{\text{rel,c}}$ and $\boldsymbol{\varepsilon_g}$ highlights that the augmented Lagrangian framework is able to quickly compute a solution with moderate accuracy. When further tightening the tolerances, the computation time as well as the required iterations increase clearly. This is to be expected as augmented Lagrangian and gradient algorithms are linearly convergent opposed to super-linear or quadratic convergence of, e.g., SQP or interior point methods. However, as the main application of GRAMPC is model predictive control, the ability to quickly compute suboptimal solutions that are improved over time is more important than the numerical solution for very small tolerances.

The resulting trajectory of the planar robot is depicted in Fig. 12 and shows that the solution of the optimal control problem (49a) involves moving through singular configurations of left and right arm, which makes this problem quite challenging.

5.2.5 Moving horizon estimation

Another application domain for GRAMPC is moving horizon estimation (MHE) by taking advantage of the parameter optimization functionality. This is illustrated for the CSTR reactor model listed in the MPC testbench in Table 1. The system dynamics of the reactor is given by Rothfuss et al. (1996)

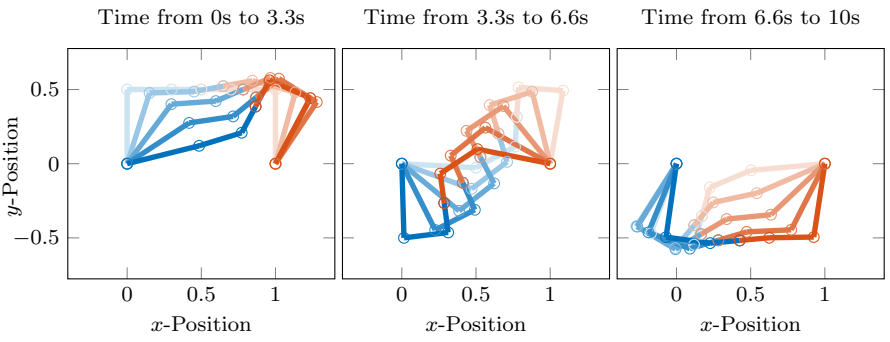

Fig. 12 Solution trajectory for the planar two-arm robot with closed kinematics

$$\dot{c}_\mathrm{A} = -k_1(T)c_\mathrm{A} - k_2(T)c_\mathrm{A}^2 + (c_\mathrm{in} - c_\mathrm{A})u_1 \tag{50a}$$

$$\dot{c}_\mathrm{B} = k_1(T)c_\mathrm{A} - k_1(T)c_\mathrm{B} - c_\mathrm{B}u_1 \tag{50b}$$

$$\begin{aligned}\dot{T} &= -\delta(k_1(T)c_\mathrm{A}\Delta H_\mathrm{AB} + k_1(T)\Delta H_\mathrm{BC} + k_2(T)c_\mathrm{A}^2\Delta H_\mathrm{AD}) \\ &\quad + \alpha(T_\mathrm{C} - T) + (T_\mathrm{in} - T)u_1\end{aligned} \tag{50c}$$

$$\dot{T}_\mathrm{C} = \beta(T - T_\mathrm{C}) + \gamma u_2 \tag{50d}$$

with the state vector $\boldsymbol{x} = [c_\mathrm{A}, c_\mathrm{B}, T, T_\mathrm{C}]^\mathrm{T}$ consisting of the monomer and product concentrations c_A, c_B and the reactor and cooling temperature T and T_C. The control variables $\boldsymbol{u} = [u_1, u_2]^\mathrm{T}$ are the normalized flow rate and cooling power. The measured quantities are the temperatures $y_1 = T$ and $y_2 = T_\mathrm{C}$. The parameter values and a more detailed description of the system are given in Rothfuss et al. (1996).

The following scenario considers the interconnection of the MHE with the MPC from the testbench, i.e. the state $\hat{\boldsymbol{x}}_k$ at the sampling time t_k is estimated and provided to the MPC. The cost functional of the MPC is designed according to

$$J(\boldsymbol{u};\boldsymbol{x}_k) := \Delta\boldsymbol{x}^\mathrm{T}(T)\boldsymbol{P}\Delta\boldsymbol{x}(T) + \int_0^T \Delta\boldsymbol{x}^\mathrm{T}(t)\boldsymbol{Q}\Delta\boldsymbol{x}(t) + \Delta\boldsymbol{u}^\mathrm{T}(t)\boldsymbol{R}\Delta\boldsymbol{u}(t)\,\mathrm{d}t\,, \tag{51}$$

where $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_\mathrm{des}$ and $\Delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}_\mathrm{des}$ penalize the deviation of the state and control from the desired setpoint $(\boldsymbol{x}_\mathrm{des}, \boldsymbol{u}_\mathrm{des})$. The MHE uses the cost functional defined in Sect. 2.3, c.f. (5a). The sampling rate for both MPC and MHE is set to $\Delta t = 1$ s. The MPC runs with a prediction horizon of $T = 20$ min and 40 discretization points,

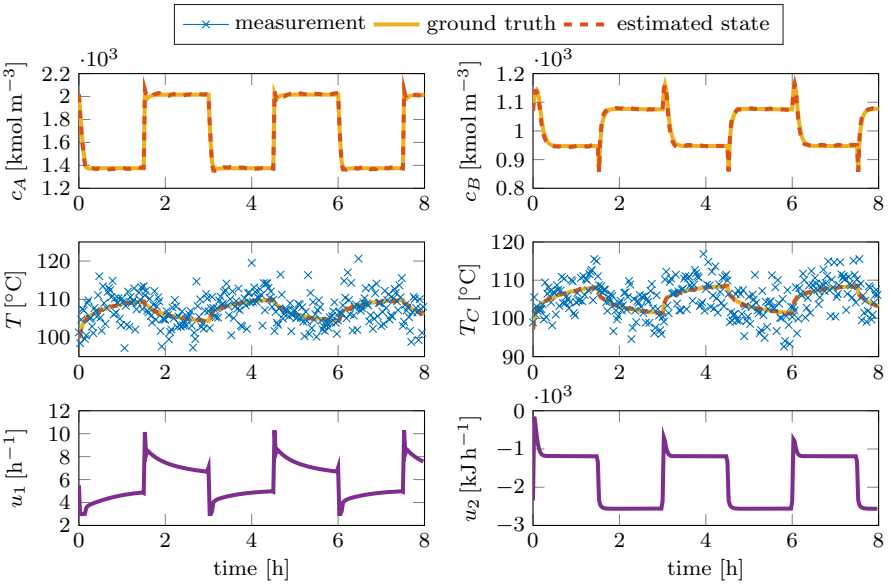

Fig. 13 MHE/MPC simulation results for the CSTR reactor

while the MHE horizon is set to $T_{\mathrm{MHE}} = 10\,\mathrm{s}$ with 10 discretization points. The GRAMPC implementation of the MHE uses only one gradient iteration per sampling step, i.e. $(i_{\max}, j_{\max}) = (1, 1)$, while the implementation of the MPC uses three gradient iterations, i.e. $(i_{\max}, j_{\max}) = (1, 3)$.

The simulated scenario in Fig. 13 consists of alternating setpoint changes between two stationary points. The two measured temperatures are subject to a Gaussian measurement noise with a standard deviation of 4 °C. The initial guess for the state vector $\boldsymbol{x}$ of the MHE differs from the true initial values by 100 kmol m^{-3} for both concentrations c_{A} and c_{B} and by 5 °C, respectively −7 °C, for the reactor and cooling temperature. This initial disturbance vanishes within a few iterations and the MHE tracks the ground truth (i.e. the simulated state values) closely, with an average error of $\delta\hat{\boldsymbol{x}} = [7.12\ \mathrm{kmol\ m^{-3}}, 6.24\ \mathrm{kmol\ m^{-3}}, 0.10\ °\mathrm{C}, 0.09\ °\mathrm{C}]^{\mathsf{T}}$. This corresponds to a relative error of less than 0.1% for each state variable, when normalized to the respective maximum value. One iteration of the MHE requires a computation time of $t_{\mathrm{CPU}} = 11\,\mu\mathrm{s}$ to calculate a new estimate of the state vector $\boldsymbol{x}$ and therefore about one third of the time necessary for one MPC iteration.

6 Conclusions

The augmented Lagrangian algorithm presented in this paper is implemented in the nonlinear model predictive control (MPC) toolbox GRAMPC and extends its original version that was published in 2014 in several significant aspects. The system class that can be handled by GRAMPC are general nonlinear systems described by explicit or semi-implicit differential equations or differential-algebraic equations (DAE) of index 1. Besides input constraints, the algorithm accounts for nonlinear state and/or input dependent equality and inequality constraints as well as for unknown parameters and a possibly free end time as further optimization variables, which is relevant, for instance, for moving horizon estimation or MPC on a shrinking horizon. The computational efficiency of GRAMPC is illustrated for a testbench of representative MPC problems and in comparison with two state-of-the-art nonlinear MPC toolboxes. In particular, the augmented Lagrangian algorithm implemented in GRAMPC is tailored to embedded MPC with very low memory requirements. This is demonstrated in terms of runtime results on dSPACE and ECU hardware that is typically used in automotive applications. GRAMPC is available at http://sourceforge.net/projects/grampc and is licensed under the GNU Lesser General Public License (version 3), which is suitable for both academic and industrial/commercial purposes.

Appendix: Transformation of inequality to equality constraints

This appendix describes the transformation of the inequality constraints (1d) to the equality constraints (10) in more detail. A standard approach of augmented Lagrangian methods is to introduce slack variables $\boldsymbol{v} \geq 0$ and $\boldsymbol{v}_T \geq 0$ in order to add (1d) to the existing equality constraints (1c) according to

$$\hat{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{v}_T) = \begin{bmatrix} \boldsymbol{g}_T(\boldsymbol{x},\boldsymbol{p},T) \\ \boldsymbol{h}_T(\boldsymbol{x},\boldsymbol{p},T) + \boldsymbol{v}_T \end{bmatrix} = \boldsymbol{0}\,, \quad \hat{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{v}) = \begin{bmatrix} \boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t) \\ \boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t) + \boldsymbol{v} \end{bmatrix} = \boldsymbol{0}\,. \tag{52}$$

The set of constraints (52) are then adjoined to the cost functional (1a)

$$\hat{J}(\boldsymbol{u},\boldsymbol{p},T,\boldsymbol{\mu},\boldsymbol{\mu}_T,\boldsymbol{c},\boldsymbol{c}_T,\boldsymbol{v},\boldsymbol{v}_T;\boldsymbol{x}_0) = \hat{V}(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_T,\boldsymbol{c}_T,\boldsymbol{v}_T) + \int_0^T \hat{l}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{v})\,\mathrm{d}t \tag{53}$$

with the new terminal and integral cost functions

$$\hat{V}(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{\mu}_T,\boldsymbol{c}_T,\boldsymbol{v}_T) = V(\boldsymbol{x},\boldsymbol{p},T) + \boldsymbol{\mu}_T^{\mathsf{T}}\hat{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{v}_T) + \frac{1}{2}\left\|\hat{\boldsymbol{g}}_T(\boldsymbol{x},\boldsymbol{p},T,\boldsymbol{v}_T)\right\|_{\boldsymbol{c}_T}^2 \tag{54a}$$

$$\hat{l}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t,\boldsymbol{\mu},\boldsymbol{c},\boldsymbol{v}) = l(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t) + \boldsymbol{\mu}^{\mathsf{T}}\hat{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{p},\boldsymbol{u},t,\boldsymbol{v}_T) + \frac{1}{2}\left\|\hat{\boldsymbol{g}}(\boldsymbol{x},\boldsymbol{p},\boldsymbol{u},t,\boldsymbol{v}_T)\right\|_{\boldsymbol{C}}^2 \tag{54b}$$

using the multipliers $\boldsymbol{\mu}$ and $\boldsymbol{\mu}_T$, the penalties $\boldsymbol{c}$ and $\boldsymbol{c}_T$ and the corresponding diagonal matrices $\boldsymbol{C}$ and $\boldsymbol{C}_T$. Instead of solving the original OCP (1), the augmented Lagrangian approach solves the max–min-problem

$$\max_{\boldsymbol{\mu},\boldsymbol{\mu}_T} \min_{\boldsymbol{u},\boldsymbol{p},T,\boldsymbol{v},\boldsymbol{v}_T} \quad \hat{J}(\boldsymbol{u},\boldsymbol{p},T,\boldsymbol{\mu},\boldsymbol{\mu}_T,\boldsymbol{c},\boldsymbol{c}_T,\boldsymbol{v},\boldsymbol{v}_T;\boldsymbol{x}_0) \tag{55a}$$

$$\text{s.t.} \quad \boldsymbol{M}\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)\,, \quad \boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{55b}$$

$$\boldsymbol{v}(t) \geq 0\,, \quad \boldsymbol{v}_T \geq 0\,, \quad \boldsymbol{u}(t) \in [\boldsymbol{u}_{\min},\boldsymbol{u}_{\max}] \tag{55c}$$

$$\boldsymbol{p} \in [\boldsymbol{p}_{\min},\boldsymbol{p}_{\max}]\,, \quad T \in [T_{\min},T_{\max}]\,. \tag{55d}$$

The minimization of the cost functional (55a) with respect to $\boldsymbol{v}$ and $\boldsymbol{v}_T$ can be carried out explicitly. In case of $\boldsymbol{v}_T$, the minimization of (55a) reduces to the strictly convex, quadratic problem

$$\min_{\boldsymbol{v}_T \geq \boldsymbol{0}} \quad \boldsymbol{\mu}_{\boldsymbol{h}_T}^{\mathsf{T}}\left(\boldsymbol{h}_T(\boldsymbol{x},\boldsymbol{p},T) + \boldsymbol{v}_T\right) + \frac{1}{2}\left\|\boldsymbol{h}_T(\boldsymbol{x},\boldsymbol{p},T) + \boldsymbol{v}_T\right\|_{\boldsymbol{C}_{\boldsymbol{h}_T}}^2 \tag{56}$$

for which the solution follows from the stationarity condition and projection if the constraint $\boldsymbol{v}_T \geq \boldsymbol{0}$ is violated

$$\boldsymbol{v}_T = \max\left\{\boldsymbol{0}, -\boldsymbol{C}_{\boldsymbol{h}_T}^{-1}\boldsymbol{\mu}_{\boldsymbol{h}_T} - \boldsymbol{h}_T(\boldsymbol{x},\boldsymbol{p},T)\right\}. \tag{57}$$

The minimization of (55a) w.r.t. the slack variables $\boldsymbol{v} = \boldsymbol{v}(t)$ corresponds to

$$\min_{\boldsymbol{v}(\cdot)\geq\mathbf{0}} \int_0^T \boldsymbol{\mu}_{\boldsymbol{h}}^{\mathrm{T}}\left(\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)+\boldsymbol{v}\right)+\frac{1}{2}\|\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)+\boldsymbol{v}\|_{\boldsymbol{C}_{\boldsymbol{h}}}^2\,\mathrm{d}t\,. \tag{58}$$

Since $\boldsymbol{v}$ only occurs in the integral and is not influenced by the dynamics (55b), the minimization of (55a) w.r.t. $\boldsymbol{v}=\boldsymbol{v}(t)$ can be carried out pointwise in time and therefore reduces to a convex, quadratic problem similar to (56) with the pointwise solution

$$\boldsymbol{v}=\max\left\{\mathbf{0},-\boldsymbol{C}_{\boldsymbol{h}}^{-1}\boldsymbol{\mu}_{\boldsymbol{h}}-\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p},t)\right\}. \tag{59}$$

Inserting the solutions (57) and (59) into (52) eventually yields the transformed equality constraints (10).

References

Allaire G (2007) Numerical analysis and optimization. Oxford University Press, Oxford

Barzilai J, Borwein JM (1988) Two-point step size gradient methods. SIAM J Numer Anal 8(1):141–148

Bemporad A, Borrelli F, Morari M (2002a) Model predictive control based on linear programming—the explicit solution. IEEE Trans Autom Control 47(12):1974–1985

Bemporad A, Morari M, Dua V, Pistikopoulos E (2002b) The explicit linear quadratic regulator for constrained systems. Automatica 38(1):3–20

Bergounioux M (1997) Use of augmented Lagrangian methods for the optimal control of obstacle problems. J Optim Theory Appl 95(1):101–126

Berkovitz LD (1974) Optimal control theory. Springer, New York

Bertsekas DP (1996) Constrained optimization and lagrange multiplier methods. Academic Press, Belmont

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge

Cao Y, Li S, Petzold L, Serban R (2003) Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution. SIAM J Sci Comput 24(3):1076–1089

Chen H, Allgöwer F (1998) A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica 34(10):1205–1217

Conn AR, Gould G, Toint PL (1992) LANCELOT: a fortran package for large-scale nonlinear optimization (release A). Springer, Berlin

de Aguiar M, Camponogara E, Foss B (2016) An augmented Lagrangian method for optimal control of continuous time DAE systems. In: Proceedings of the IEEE conference on control applications (CCA), Buenos Aires, pp 1185–1190

Diehl M, Findeisen R, Allgöwer F, Bock H, Schlöder J (2004) Nominal stability of real-time iteration scheme for nonlinear model predictive control. IEE Proc 152(3):296–308

Diehl M, Bock H, Schlöder J (2005) A real-time iteration scheme for nonlinear optimization in optimal feedback control. SIAM J Control Optim 43(5):1714–1736

Domahidi A, Zgraggen A, Zeilinger M, Morari M, Jones C (2012) Efficient interior point methods for multistage problems arising in receding horizon control. In: Proceedings of the IEEE conference on decision and control (CDC), Maui, pp 668–674

Dunn JC (1996) On l^2 sufficient conditions and the gradient projection method for optimal control problems. SIAM J Control Optim 34(4):1270–1290

Englert T, Graichen K (2018) Nonlinear model predictive torque control of PMSMs for high performance applications. Control Eng Pract 81:43–54

Ferreau H, Bock H, Diehl M (2008) An online active set strategy to overcome the limitations of explicit MPC. Int J Robust Nonlinear Control 18(8):816–830

Ferreau H, Kirches C, Potschka A, Bock H, Diehl M (2014) qpOASES: a parametric active-set algorithm for quadratic programming. Math Program Comput 6(4):327–363

Ferreau H, Almér S, Verschueren R, Diehl M, Frick D, Domahidi A, Jerez J, Stathopoulos G, Jones C (2017) Embedded optimization methods for industrial automatic control. In: Proceedings of 20th IFAC world congress, pp 13736–13751

Findeisen R, Graichen K, Mönnigmann M (2018) Embedded optimization in control: an introduction, opportunities, and challenges. Automatisierungstechnik 66(11):877–902 (in German)

Fortin M, Glowinski R (1983) Augmented lagrangian methods: applications to the solution of boundary-value problems. North-Holland, Amsterdam

Giselsson P (2014) Improved fast dual gradient methods for embedded model predictive control. In: Proceedings of the 19th IFAC world congress, Cape Town, pp 2303–2309

Graichen K (2012) A fixed-point iteration scheme for real-time model predictive control. Automatica 48(7):1300–1305

Graichen K, Kugi A (2010) Stability and incremental improvement of suboptimal MPC without terminal constraints. IEEE Trans Autom Control 55(11):2576–2580

Graichen K, Egretzberger M, Kugi A (2010) A suboptimal approach to real-time model predictive control of nonlinear systems. Automatisierungstechnik 58(8):447–456

Grüne L (2013) Economic receding horizon control without terminal constraints. Automatica 49(3):725–734

Grüne L, Palma V (2014) On the benefit of re-optimization in optimal control under perturbations. In: Proceedings of the 21st international symposium on mathematical theory of networks and systems (MTNS), Groningen, pp 439–446

Hager W (1990) Multiplier methods for nonlinear optimal control. SIAM J Numer Anal 27(4):1061–1080

Hairer E, Wanner G (1996) Solving ordinary differential equations: stiff and differential-algebraic problems. Springer, Heidelberg

Harder K, Buchholz M, Niemeyer J, Remele J, Graichen K (2017) Nonlinear MPC with emission control for a real-world heavy-duty diesel engine. In: Proceedings of the IEEE international conference on advanced intelligent mechatronics (AIM), Munich, pp 1768–1773

Hartley E, Maciejowski J (2015) Field programmable gate array based predictive control system for spacecraft rendezvous in elliptical orbits. Optim Control Appl Methods 36(5):585–607

Houska B, Ferreau H, Diehl M (2011) ACADO toolkit—an open source framework for automatic control and dynamic optimization. Optim Control Appl Methods 32(3):298–312

Ito K, Kunisch K (1990) The augmented Lagrangian method for equality and inequality constraints in hilbert spaces. Math Program 46:341–360

Jones C, Domahidi A, Morari M, Richter S, Ullmann F, Zeilinger M (2012) Fast predictive control: real-time computation and certification. In: Proceedings of the 4th IFAC nonlinear predictive control conference (NMPC), Leeuwenhorst, pp 94–98

Kalmari J, Backman J, Visala A (2015) A toolkit for nonlinear model predictive control using gradient projection and code generation. Control Eng Pract 39:56–66

Käpernick B (2016) Gradient-based nonlinear model predictive control with constraint transformation for fast dynamical systems. Dissertation, Ulm University, Shaker, Aachen

Käpernick B, Graichen K (2013) Model predictive control of an overhead crane using constraint substitution. In: Proceedings of the american control conference (ACC), pp 3973–3978

Käpernick B, Graichen K (2014a) The gradient based nonlinear model predictive control software GRAMPC. In: Proceedings of the European control conference (ECC), Strasbourg, pp 1170–1175

Käpernick B, Graichen K (2014b) PLC implementation of a nonlinear model predictive controller. In: Proceedings of the 19th IFAC world congress, Cape Town, pp 1892–1897

Käpernick B, Süß S, Schubert E, Graichen K (2014) A synthesis strategy for nonlinear model predictive controller on FPGA. In: Proceedings of the UKACC 10th international conference on control, Loughborough, pp 662–667

Kirches C, Wirsching L, Bock H, Schlöder J (2012) Efficient direct multiple shooting for nonlinear model predictive control on long horizons. J Process Control 22(3):540–550

Kirk DE (1970) Optimal control theory: an introduction. Dover Publications, Mineola

Kouzoupis D, Ferreau H, Peyrl H, Diehl M (2015) First-order methods in embedded nonlinear model predictive control. In: Proceedings of the European control conference (ECC), Linz, pp 2617–2622

Kufoalor D, Richter S, Imsland L, Johansen T, Morari M, Eikrem G (2014) Embedded model predictive control on a PLC using a primal-dual first-order method for a subsea separation process. In: Proceedings of the 22nd mediterranean conference on control and automation (MED), Palermo, pp 368–373

Limon D, Alamo T, Salas F, Camacho E (2006) On the stability of constrained MPC without terminal constraint. IEEE Trans Autom Control 51(5):832–836
Ling K, Wu B, Maciejowski J (2008) Embedded model predictive control (MPC) using a FPGA. In: Proceedings of the 17th IFAC world congress, Seoul, pp 15250–15255
Mayne D, Rawlings J, Rao C, Scokaert P (2000) Constrained model predictive control: stability and optimality. Automatica 36(6):789–814
Mesmer F, Szabo T, Graichen K (2018) Embedded nonlinear model predictive control of dual-clutch transmissions with multiple groups on a shrinking horizon. IEEE Trans Control Syst Technol. https://doi.org/10.1109/TCST.2018.2856191
Necoara I (2015) Computational complexity certification for dual gradient method: application to embedded MPC. Syst Control Lett 81:49–56
Necoara I, Kvamme S (2015) DuQuad: a toolbox for solving convex quadratic programs using dual (augmented) first order algorithms. In: Proceedings of the IEEE conference on decision and control (CDC), Osaka, pp 2043–2048
Nedelcu V, Necoara I, Tran-Dinh Q (2014) Computational complexity of inexact gradient augmented Lagrangian methods: application to constrained MPC. SIAM J Control Optim 52(5):3109–3134
Nesterov Y (2003) Introductory lectures on convex optimization: a basic course. In: Paradalos P, Hearn D (eds) Applied optimization, applied optimization, vol 87. Kluwer Academic Publishers, Boston
Nocedal J, Wright S (2006) Numerical optimization. Springer, New York
Ohtsuka T (2004) A continuation/GMRES method for fast computation of nonlinear receding horizon control. Automatica 40(4):563–574
Richter S (2012) Computational complexity certification of gradient methods for real-time model predictive control. Ph.D. thesis ETH Zürich, ETH
Richter S, Jones C, Morari M (2012) Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. IEEE Trans Autom Control 57(6):1391–1403
Rockafellar R (1974) Augmented Lagrange multiplier functions and duality in nonconvex programming. SIAM J Control 12(2):268–285
Rothfuss R, Rudolph J, Zeitz M (1996) Flatness based control of a nonlinear chemical reactor model. Automatica 32(10):1433–1439
Sastry S (2013) Nonlinear systems: analysis, stability and control. Springer, New York
Scokaert P, Mayne D, Rawlings J (1999) Suboptimal model predictive control (feasibility implies optimality). IEEE Trans Autom Control 44(3):648–654
Skaf J, Boyd S, Zeevi A (2010) Shrinking-horizon dynamic programming. Int J Robust Nonlinear Control 20(17):1993–2002
Tøndel P, Johansen TA (2002) Complexity reduction in explicit linear model predictive control. In: Proceedings of the 15th IFAC world congress, Barcelona, pp 189–194
Utz T, Graichen K, Kugi A (2010) Trajectory planning and receding horizon tracking control of a quasilinear diffusion–convection–reaction system. In: Proceedings of the 8th IFAC symposium on nonlinear control systems, Bologna, pp 587–592
Wang Y, Boyd S (2010) Fast model predictive control using online optimization. IEEE Trans Control Syst Technol 18(2):267–278
Werling M, Reinisch P, Gresser K (2012) Kombinierte Brems-Ausweich-Assistenz mittels nichtlinearer modellprädiktiver Trajektorienplanung für den aktiven Fußgängerschutz. Tagungsband des 8. Workshop Fahrerassistenzsysteme, pp 68–77
Zienkiewicz OC, Morgan K (1983) Finite elements and approximation. Wiley, New York