Embedded Optimization Methods for Industrial Automatic Control ⋆

H.J. Ferreau [*] **S. Almér** [*] **R. Verschueren** [**] **M. Diehl** [**]
D. Frick [***] **A. Domahidi** [****] **J.L. Jerez** [****] **G. Stathopoulos** [†]
C. Jones [†]

[*] *ABB Corporate Research, Segelhofstrasse 1K, 5405 Baden-Dättwil, Switzerland, e-mail: {joachim.ferreau,stefan.almer}@ch.abb.com*
[**] *Systems Control and Optimization Laboratory, Department of Microsystems Engineering and Department of Mathematics, University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany, e-mail: {robin.verschueren,moritz.diehl}@imtek.uni-freiburg.de*
[***] *Automatic Control Laboratory, ETH Zürich, Physikstrasse 3, 8092 Zurich, Switzerland, e-mail: dafrick@control.ee.ethz.ch*
[****] *embotech GmbH, Physikstrasse 3, Automatic Control Laboratory, 8092 Zurich, Switzerland, e-mail: {domahidi,jerez}@embotech.com*
[†] *Automatic Control Laboratory, École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland, e-mail: {georgios.stathopoulos,colin.jones}@epfl.ch*

Abstract:
Starting in the late 1970s, optimization-based control has built up an impressive track record of successful industrial applications, in particular in the petrochemical and process industries. More recently, optimization methods for automatic control are more and more deployed on so-called embedded hardware to cater for application-specific needs such as guaranteed communication latency, low energy consumption or cost effectiveness. This development greatly broadens the scope of applications to which optimization methods can be applied to sectors such as robotics, automotive, aerospace or power electronics. However, it also poses additional challenges regarding both the algorithmic concepts and their actual implementations for a given computing hardware. This survey paper discusses key challenges for using embedded optimization methods and summarizes their main use cases in current industrial practice. Motivated by this discussion, a number of dedicated embedded optimization algorithms and their actual implementations are reviewed. The presentation is organized according to the mathematical structure of the embedded optimization problem, ranging from convex quadratic programming over more general convex and nonconvex problems to formulations comprising discrete optimization variables.

Keywords: Optimization; Optimal control; Automatic Control; Model-based control; Predictive control; Real-time systems; Embedded systems; Industrial control

1. BACKGROUND

1.1 Historical remarks

When the first programmable digital computers appeared in the 1940s, mathematicians and engineers quickly adopted this new technology for the numerical solution of mathematical optimization problems. Arguably the most prominent example is the simplex algorithm for minimizing a linear cost function subject to linear constraints (Dantzig (1963)). The subsequent exponential growth in computing power (per dollar or square centimeter of silicon) enabled the solving of ever larger and more difficult problems, and control engineers started to consider microprocessors a viable option for process control.

⋆ Support by the EU via ERC-HIGHWIND (259 166), ITN-TEMPO (607 957), and ITN-AWESCO (642 682) and by the DFG within Reseach Unit FOR 2401 is gratefully acknowledged.

Starting in the late 1970s, model predictive control (MPC) has emerged as a successful symbiosis of numerical optimization and feedback control theory, see e.g. Richalet et al. (1978); García et al. (1989); Mayne et al. (2000); Rawlings and Mayne (2009). Based on a dynamic model of the process to be controlled, MPC optimizes the predicted future process behavior by choosing the best admissible control actions. The past three decades have witnessed thousands of successful applications of MPC in the process industry, where the optimizer typically solves a constrained linear-quadratic MPC problem every couple of seconds or minutes on industrial PCs, see the surveys by Qin and Badgwell (2003); Darby and Nikolaou (2012).

More recently, researchers achieved spectacular advances in both theory and algorithms for solving optimization problems in real-time on embedded controller hardware. For example, MPC has been implemented on embed-

ded CPUs (see e.g. Ferreau et al. (2007); Vukov et al. (2012); Liniger et al. (2015)), digital signal processors (see e.g. Wills et al. (2005); Almér et al. (2013); Beccuti et al. (2009)), programmable logic controllers (see e.g. Huyck et al. (2012)), or field-programmable gate arrays (see e.g. Mönnigmann and Kastsian (2011); Hartley and Maciejowski (2010); Boechat et al. (2013); Jerez et al. (2014)) at up to Megahertz sampling rates.

1.2 What does "embedded" optimization mean?

An "embedded system" typically refers to a computer system that is an integrated part of a larger, complete device. This definition implies little about whether a numerical optimization can be carried out on an embedded system or not. Consequently, the notion "embedded" is often used in a rather vague manner within the optimization and automatic control community. In order to define the term "embedded optimization" and to clarify the scope of this survey, a few related notions will be reviewed first:

Does "embedded" mean *fast*? If this notion is used to express that some computation shall be carried out within short time periods, it hardly contributes to a useful definition. A priori there is no maximum time span for performing an optimization to be called "embedded"; why should embedded optimization not run at a sampling rate of one minute or more? Moreover, mentioning execution times without considering dimensions and other characteristics of the optimization problem at hand is rather meaningless. Embedded optimization may solve large-scale, complex problems. If one talks about *efficient* rather than fast optimization, this becomes somewhat more descriptive as embedded optimization is typically performed on computing hardware with limited resources that need to be used with care.

So, does "embedded" mean *limited resources*? In our experience, most often this is the case. Main reasons for using embedded computing hardware are reduced hardware cost or size (compared to, e.g., a desktop PC) and reduced power consumption of the device. Consequently, embedded optimization methods shall be deployable on hardware with limited computational power or memory, and possibly even reduced accuracy for representing numbers within the calculations. However, if that would be all to it, a resource-efficient implementation for solving large-scale optimization problems on a computer cluster installed in the hull of a container ship should be called "embedded optimization", too.

Does "embedded" mean *real-time*? This is indeed the most important requirement from an algorithmic perspective; whatever a computation is carried out, it needs to terminate within a pre-defined maximum time limit returning a sufficiently accurate solution. But why is this important?

Because "embedded" also means *autonomous*. If there is always a human operator who can block the result of the optimization algorithm and enforce alternative input signals to the system, we would be very reluctant to call this optimization "embedded". This also creates a connection to the first notion: if the optimization is carried out too frequently for a human operator to supervise the outcome, it needs to run autonomously. However, also slow computations can be setup autonomously. In the container ship example, if the cluster runs for minutes obeying a strict real-time limit and the result is used without supervision by a human operator, then (and only then) one may call even this setup embedded optimization.

The above considerations can be summarized in the following two informal definitions:

Definition 1. An optimization method is called *embedded* if and only if it a) can run autonomously, i.e., if it delivers a sufficiently accurate solution within a real-time limit given a priori, and b) can be deployed on a computing hardware with limited resources.

Definition 2. An optimization application is called *embedded* if and only if an embedded optimization method is run autonomously on a computing hardware that is an integrated part of a larger, complete device and its outcome is used to influence the behaviour of the device.

1.3 Industry needs

In order for (embedded) optimization to be a viable choice for *industrial* automatic control, a number of general requirements need to be met. We briefly discuss a couple of important ones.

Reliability: According to Definition 2, the embedded optimization algorithm needs to run autonomously and is an integral part of a larger device, typically a commercial product. Any failure of such a device may lead to serious financial damage or even endanger the health of people. It is therefore of uttermost importance that the optimization algorithm can be implemented in a highly reliable manner. If a failure of the algorithm cannot be excluded, such a situation should at least be reliably detected and mitigated by means of a suitable backup strategy.

Restricted hardware: As customers are often price-sensitive, this implies a pressure to use the least expensive computational hardware that meets the requirements. Moreover, the choice of the hardware might be pre-determined by other considerations regarding the product. This favours optimization algorithms that are efficient, allow for a lightweight implementation and offer "tuning knobs" to trade-off computational load against solution accuracy.

Rapid prototyping: With product development cycles becoming shorter, there is also less time to setup and tune the underlying optimization algorithm. This favors algorithms that are generally applicable and do not rely on some kind of "magic" to make them work well on a specific problem. Even if these procedures can be automated, one should note that any kind of time-consuming preprocessing (which in a broader sense also includes code generation or explicit pre-solving of the problem) makes prototyping less agile.

Maintainability: Scientists may write their PhD thesis and leave, products may need to be maintained for decades. Algorithms that are easier to understand and give clear indications of why they failed in a particular situation are more likely to be successfully maintained by non-academic experts and thus more attractive to be used within a commercial product.

1.4 Outline

The remainder of this paper is organized as follows: Section 2 describes important use cases of embedded optimization in industrial applications. These use cases are then categorized by their mathematical structure in Section 3. Based on this categorization, different embedded optimization algorithms and implementations are discussed in the following sections. Namely, Section 4 discusses algorithms for convex quadratic programming. Section 5 addresses algorithms for general convex optimization, with a a focus on splitting methods. Section 6 and Section 7 review algorithms for nonconvex and discrete optimization, respectively. Section 8 concludes the paper.

2. USE CASES

Embedded optimization may be used in different ways. This section introduces its major use cases and also mentions a couple of examples of successful applications of embedded optimization from various fields. Note that this paper focuses on methods and implementations for embedded optimization and that a complete survey on actual applications is beyond its scope. The mathematical structure of the embedded optimization problems to be solved is discussed in detail in Section 3.

2.1 Model predictive control

Optimization algorithms are only deployed on embedded hardware if they are to run repeatedly. A common reason for doing so is optimization-based control, typically in form of model predictive control (MPC). MPC is a feedback control approach that repeatedly solves optimization problems at a fixed sampling rate (see Camacho and Bordons (2007); Rawlings and Mayne (2009)). For a given application, these problems are structurally identical but differ in certain parameters that incorporate the feedback information into the problem. While MPC has traditionally been used mainly in the process and petrochemical industry (see Qin and Badgwell (2003)), embedded MPC is becoming frequently used in industrial domains such as automotive, robotics, aerospace, power electronics or the energy sector.

MPC problems solved in automotive applications are typically of linear-quadratic type and require sampling times in the order of 10–100 milliseconds. For example, linear MPC has been used in experiments for Diesel engine air path control (Ortner and del Re (2007)), combustion control in a HCCI engine (Widd et al. (2009)), idle speed control (Di Cairano et al. (2012)) or hybrid electric vehicle energy management (Di Cairano et al. (2013)).

A particularly challenging domain is power electronics as the nature of the control problems requires both explicit treatment of nonlinearities or discrete decisions and very fast sampling times in the order of 20–1000 microseconds. MPC involving discrete decisions based on explicit approaches have been used for controlling a voltage source inverter with LCL filter (Mariethoz and Morari (2009) and Almér et al. (2013)) or a step-down DC-DC converter (Geyer et al. (2008)). Nonlinear MPC formulations tackled with iterative solution methods have been used to control, e.g., load commutated inverters (see Besselmann et al. (2016b,a)) or a voltage source inverter with LCL filter (Almér et al. (2015)).

There exist many more embedded MPC applications and we restrict ourselves to mention the following, just to illustrate how broad the range of possible applications has become: early experiments using MPC for controlling a hovercraft has been reported in Seguchi and Ohtsuka (2003b), Noga et al. (2014) describes experimental results on using nonlinear MPC for cryogenic cooling control, an MPC based on a mixed-integer quadratic programming formulation for crane control has been proposed in Iles et al. (2014) and van Duijkeren et al. (2016) employ path-following MPC for serial-link robot manipulators.

2.2 Moving horizon estimation

Embedded optimization is also used for estimating the current state of the process to be controlled. If the process exhibits substantially nonlinear dynamics, moving horizon estimation (MHE) may be the preferred choice for obtaining accurate estimates. Similar to MPC, also MHE requires one to solve sequences of optimization problems whose mathematical structure is fixed but the parameters representing the actual measurements vary over time.

Embedded MHE has been demonstrated to deliver superior practical performance in a number of applications domains. For example, it has been applied to autonomous agricultural vehicles (Kraus et al. (2013)), to overhead cranes (see Debrouwere et al. (2014) and Vukov et al. (2015)), to induction machines (Frick et al. (2012)) or to an active cantilever (Abdollahpouri et al. (2017)). In these applications, embedded MHE was running at sampling rates of up to 4 kHz.

2.3 Static optimization, scheduling and planning

The use of embedded optimization has also been investigated beyond the control domain. For example, Vargas-Villamil and Rivera (2000) used it for scheduling of semiconductor manufacturing lines, and Biegler and Zavala (2009) present a framework for enterprise-wide dynamic optimization. An industrial perspective on integrating production scheduling and control is given in Harjunkoski et al. (2009). An example of embedded static optimization is given by Defraene et al. (2012), where a convex optimization problem is solved repeatedly for improved clipping of audio signals.

3. MATHEMATICAL STRUCTURE

Optimization problems arising in embedded applications almost always feature a deterministic formulation (in contrast to stochastic optimization), a single objective function to be optimized (in contrast to multi-objective optimization) and sparse problem data (e.g. in multistage problems). This section further categorizes such deterministic, single-objective optimization problems according to their mathematical structure. This is an important prerequisite for discussing embedded solution methods in subsequent sections as the problem structure strongly influences both their respective numerical efficiency and their reliability.

3.1 Convex optimization

We state general convex optimization problems in the following form:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & f(w) && \text{(1a)} \\ \text{subject to} \quad & Cw = \overline{c}, && \text{(1b)} \\ & w \in \Omega, && \text{(1c)} \end{aligned}$$

where we define the convex function $f : \mathbb{R}^n \to \mathbb{R}$, the matrix $C \in \mathbb{R}^{n_{\text{eq}} \times n}$, the vector $\overline{c} \in \mathbb{R}^{n_{\text{eq}}}$ and the convex set Ω. Convex optimization problems constitute an important subclass as any *locally optimal solution* w^* (i.e. $f(w^*) \leq f(w)$ for all feasible w in a neighborhood around w^*) is also a *global solution* (i.e. $f(w^*) \leq f(w)$ for all feasible w) . This strongly reduces the likelihood of iterative methods getting stuck at sub-optimal "solutions".

Quadratic programming An important subclass of convex optimization problems are convex quadratic programming (QP) problems, which can be written in different forms equivalent to the following one:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2} w^\top H w + g^\top w && \text{(2a)} \\ \text{subject to} \quad & Cw = \overline{c}, && \text{(2b)} \\ & Dw \leq \overline{d} && \text{(2c)} \end{aligned}$$

with positive semi-definite Hessian matrix $H \in \mathbb{R}^{n \times n}$, gradient vector $g \in \mathbb{R}^n$, constraint matrices $C \in \mathbb{R}^{n_{\text{eq}} \times n}$, $D \in \mathbb{R}^{n_{\text{iq}} \times n}$ and right-hand side vectors $\overline{c} \in \mathbb{R}^{n_{\text{eq}}}$, $\overline{d} \in \mathbb{R}^{n_{\text{iq}}}$. We associate the Lagrange multipliers $\lambda \in \mathbb{R}^{n_{\text{eq}}}$ with the equality constraints and the multipliers $\mu \in \mathbb{R}^{n_{\text{iq}}}$ with the inequalities. The vector of dual variables will be denoted by $y = [\lambda^\top, \mu^\top]^\top$. If the Hessian matrix is set to zero, one gets a *linear programming (LP)* problem.

Linear optimal control Linear MPC is an optimal control method for systems governed by linear dynamics:

$$x_{k+1} = Ax_k + Bu_k, \tag{3}$$

where we introduced discrete-time state vectors $x_k \in \mathbb{R}^{n_x}$ and control inputs $u_k \in \mathbb{R}^{n_u}$, as well as the system matrix $A \in \mathbb{R}^{n_x \times n_x}$ and the control matrix $B \in \mathbb{R}^{n_x \times n_u}$, respectively. Linear MPC repeatedly optimizes the predicted behavior of the system along a finite horizon and therefore solves at each sampling instant a specially structured convex QP problem:

$$\begin{aligned} \underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + x_N^\top P x_N \\ \text{subject to} \quad & x_{k+1} = Ax_k + Bu_k,\ k = 0, \dots, N-1, \\ & D_k^x x_k + D_k^u u_k \leq \overline{d}_k,\ k = 0, \dots, N-1, \\ & D_N^x x_N \leq \overline{d}_N, \\ & x_0 = \overline{x}_0. \end{aligned} \tag{4}$$

The positive semi-definite weighting matrices are denoted by $Q, P \in \mathbb{R}^{n_x \times n_x}$, $S \in \mathbb{R}^{n_u \times n_x}$ and $R \in \mathbb{R}^{n_u \times n_u}$, whereas the inequality constraint matrices are $D_k^x \in \mathbb{R}^{n_{\text{iq},k} \times n_x}, D_k^u \in \mathbb{R}^{n_{\text{iq},k} \times n_u}$, for states and controls, respectively, together with the vector $\overline{d} \in \mathbb{R}^{n_{\text{iq},k}}$. At a given sampling instant, the initial value for the state is assumed to be a fixed vector $\overline{x}_0 \in \mathbb{R}^{n_x}$.

3.2 Non-convex optimization

Continuous optimization problems that are known (or at least not required) to be non-convex are usually expressed as nonlinear programming (NLP) problems:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & f(w) && \text{(5a)} \\ \text{subject to} \quad & c(w) = 0, && \text{(5b)} \\ & d(w) \leq 0 && \text{(5c)} \end{aligned}$$

with f as in (1) but possibly non-convex. Furthermore, we define functions $c : \mathbb{R}^n \to \mathbb{R}^{n_{\text{eq}}}$ and $d : \mathbb{R}^n \to \mathbb{R}^{n_{\text{iq}}}$. With each of the constraint functions, we associate *Lagrange multipliers* λ, μ, as in (2), such that the definition of the Lagrangian function reads as

$$\mathcal{L}(w, \lambda, \mu) := f(w) + \lambda^\top c(w) + \mu^\top d(w)\,. \tag{6}$$

Under the common assumption, that f, g, h are smooth and that LICQ holds at the solution (Nocedal and Wright, 2006, Ch. 12), the following equations form the first order necessary conditions for optimality, and are well-known as the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} \nabla_w \mathcal{L}(w, \lambda, \mu) &= 0, && \text{(7a)} \\ c(w) &= 0, && \text{(7b)} \\ d(w) &\leq 0, && \text{(7c)} \\ \mu &\geq 0, && \text{(7d)} \\ \mu^\top d(w) &= 0. && \text{(7e)} \end{aligned}$$

Nonlinear optimal control When reviewing methods for solving NLP problems, the focus is on formulations that arise from nonlinear *optimal control problems (OCP)*. We consider OCPs that are NLP problems with a particular stage-wise structure (e.g. obtained through a multiple shooting discretization of a continuous-time OCP, see Bock and Plitt (1984)):

$$\begin{aligned} \underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} l(x_k, u_k) + E(x_N) && \text{(8a)} \\ \text{subject to} \quad & x_{k+1} = F_k(x_k, u_k), \quad k = 0, \dots, N-1, && \text{(8b)} \\ & d_k(x_k, u_k) \leq 0, \quad k = 0, \dots, N-1, && \text{(8c)} \\ & d_N(x_N) \leq 0, && \text{(8d)} \\ & x_0 = \overline{x}_0. && \text{(8e)} \end{aligned}$$

We define the discrete-time dynamic system equations as $F_k : \mathbb{R}^{n_x + n_u} \to \mathbb{R}^{n_x}$, typically obtained through numerical integration of continuous-time dynamic equations. We denote the number of inequality constraints in each stage k by $n_{\text{iq},k}$, such that we can write the inequality constraints as functions $d_k : \mathbb{R}^{n_x + n_u} \to \mathbb{R}^{n_{\text{iq},k}}$. To simplify notation, we gather all optimization variables of the OCP in the vector $w = [x_0^\top, u_0^\top, x_1^\top, \dots, x_N^\top]^\top$.

We observe that the Lagrangian function, as defined in (6), is partially separable for problems with OCP structure (8). More specifically, the Hessian of the Lagrangian has the following form:

$$\nabla_w^2 \mathcal{L}(\overline{w}, \overline{\lambda}, \overline{\mu}) = \begin{bmatrix} Q_0 & S_0^\top & & & \\ S_0 & R_0 & & & \\ & & Q_1 & S_1^\top & \\ & & S_1 & R_1 & \\ & & & & \ddots \\ & & & & & Q_N \end{bmatrix}, \tag{9}$$

where we used the shorthand notation $Q_k := \nabla^2_{x_k}\mathcal{L}$, $S_k := \nabla^2_{u_k,x_k}\mathcal{L}$, $R_k := \nabla^2_{u_k}\mathcal{L}$, where each matrix is evaluated at the point $(\overline{w}, \overline{\lambda}, \overline{\mu})$. This observation is important with respect to the efficiency of numerical methods to solve nonlinear OCPs, as we will see in Section 6.

Alternatives to multiple shooting for discretizing a continuous-time OCP are so-called *single shooting* (see Sargent and Sullivan (1978)), which however is not suitable for unstable dynamics, and *direct collocation* (see Biegler (1984)), which is useful for large-scale applications, as the subproblems contain even more structure than (8).

Receding horizon control Nonlinear MPC consists of continuously solving OCPs of the form (8) in order to control the system in an "optimal" fashion. The discrete-time model (8b) is used to predict the future system behavior and consequently to optimize for the states x_k and controls u_k. Once the solution is found, only the first optimal control value u_0 is applied to the system at each time step and the procedure is repeated.

In (nonlinear) MPC, it is often the case that the desired control performance is to track some reference signal closely. Typically, this is expressed with least-squares terms in the objective function (8a):

$$\begin{aligned} f_{\text{LS}}(w) = \frac{1}{2}\sum_{k=0}^{N-1} \|h_k(x_k,u_k) - z_k\|_2^2 \\ + \frac{1}{2}\|h_N(x_N) - z_N\|_2^2, \end{aligned} \tag{10}$$

where $h_k : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_r}$, for $k = 0,\dots,N-1$, and $h_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_r}$ are nonlinear output functions that should track the reference signal $z_k \in \mathbb{R}^{n_r}, k = 0,\dots,N$.

Moving horizon estimation A related problem is moving horizon estimation (MHE), where instead one looks a fixed number of time steps into the past, using recorded measurements to optimize for an optimal estimate for the current state of the system $\overline{x}_0$. The "controls" for the case of MHE take the form of process noise. The fact that the value $\overline{x}_0$ is not fixed forms the biggest structural difference with MPC.

3.3 Optimization with discrete decisions

Discrete decisions are part of many industrial control applications. Convex problems of the form (1), or nonconvex problems of form (5) where some or all of the decision variables w are restricted to the set of integers $\mathbb{Z}$ are of particular interest. These problems are called *mixed-integer nonlinear programming (MINLP)* problems and take the following form:

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_c}\times\mathbb{Z}^{n_i}}{\text{minimize}} \quad & f(w) && \text{(11a)}\\ \text{subject to} \quad & c(w) = 0, && \text{(11b)}\\ & d(w) \le 0 && \text{(11c)} \end{aligned}$$

with $n := n_c + n_i$ and f, c, d are defined analogously to their smooth counterparts in (5).

Mixed-integer quadratic programming When the cost of a MINLP problem is restricted to be linear (or convex quadratic), and the constraints are polyhedral, these classes of problems are called mixed-integer linear (or quadratic) programming (MILP/MIQP) problems; defined as

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_c}\times\mathbb{Z}^{n_i}}{\text{minimize}} \quad & \frac{1}{2}w^\top H w + g^\top w && \text{(12a)}\\ \text{subject to} \quad & Cw = \overline{c}, && \text{(12b)}\\ & Dw \le \overline{d} && \text{(12c)} \end{aligned}$$

with $n := n_c + n_i$, positive semi-definite Hessian matrix $H \in \mathbb{R}^{n\times n}$, and $g, C, \overline{c}, D, \overline{d}$ are analogue to (2). In practice, many decisions are binary, or can be modeled as a combination of few binary decisions. For these problems, called *mixed-binary linear or quadratic (MBP)*, $\mathbb{Z}^{n_i}$ is replaced by $\{0,1\}^{n_i}$. In the control literature, most problems referred to as MILP/MIQP problems only contain binary decision variables and we will therefore use these terms interchangeably.

Optimal control with discrete decisions Discrete-time hybrid systems can be represented, as discrete hybrid automata (DHA), piece-wise affine (PWA) systems, mixed-logical dynamical (MLD) systems, linear complementarity (LC) systems, and more. Under some mild assumptions, all of these descriptions are equivalent and one can be transformed into the other, see Heemels et al. (2001); Torrisi and Bemporad (2004). The MLD framework (Bemporad and Morari (1999)) is particularly useful from a computational perspective, because it represents the hybrid dynamics as a system of mixed-integer linear constraints, by introducing auxiliary binary and continuous variables $\delta_k \in \{0,1\}^{n_\delta}$ and $z_k \in \mathbb{R}^{n_z}$:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_\delta\delta_k + B_z z_k + \overline{b}, && \text{(13a)}\\ E_x x_k + E_u u_k + E_\delta\delta_k + E_z z_k &\le \overline{e}, && \text{(13b)} \end{aligned}$$

where some of the state vectors $x_k \in \mathbb{R}^{n_{x,c}} \times \{0,1\}^{n_{x,b}}$ and controls $u_k \in \mathbb{R}^{n_{u,c}} \times \{0,1\}^{n_{u,b}}$ can also be binary. The matrices $B_\delta, B_z, E_x, E_u, E_\delta, E_z$ and the vector $\overline{b}$ are of appropriate dimension, with $\overline{e} \in \mathbb{R}^{n_{iq}}$. In the context of model-predictive control, this gives rise to a multistage MBP formulation for mixed-integer MPC problems:

$$\begin{aligned} \underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1},\\ \delta_0,\dots,\delta_{N-1},\\ z_0,\dots,z_{N-1}}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q & S^\top \\ S & R \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + x_N^\top P x_N \\ \text{subject to} \quad & x_k \in \mathbb{R}^{n_{x,c}} \times \{0,1\}^{n_{x,b}}, \\ & u_k \in \mathbb{R}^{n_{u,c}} \times \{0,1\}^{n_{u,b}}, \\ & \delta_k \in \{0,1\}^{n_\delta}, z_k \in \mathbb{R}^{n_z}, \\ & x_{k+1} = Ax_k + Bu_k + B_\delta\delta_k + B_z z_k + \overline{b}, \\ & E_x x_k + E_u u_k + E_\delta\delta_k + E_z z_k \le \overline{e}, \\ & D_N^x x_N \le \overline{d}_N, \\ & x_0 = \overline{x}_0. \end{aligned} \tag{14}$$

The matrices Q, P, S, D_N^x are as in (4). Standard polyhedral state and input constraints can be added to this formulation without loss of generality.

4. METHODS FOR CONVEX QUADRATIC PROGRAMMING PROBLEMS

4.1 Overview

The first methods for solving convex QP problems emerged from adaptations of the famous simplex method for solving

LPs (Dantzig, 1963). These so-called *active-set methods* still play an important role in solving QP problems in embedded applications, mainly due to the efficiency of existing implementations and the fact that so-called *warm-starting* (i.e. reusing solution information of the previous problem instance) may greatly reduce computational load. However, from a theoretical perspective, they lack practical a priori bounds on the number of iterations that they may need to find a solution. Historically, that has been a major motivation to develop *interior-point* (sometimes also called *path-following*) *methods*, for which polynomial runtime guarantees have been proved in 1984 for the first time (see Karmarkar (1984)). Nowadays, interior-point methods are often a preferred choice in embedded quadratic programming due to their rather constant computational load when solving various instances of related problems. Moreover, they allow for exploiting the inherent sparsity of optimal control problems as sketched in Section 3 in a straight-forward manner.

Both these two traditional approaches for solving convex QP problems suffer from common drawbacks that have inspired the quest for alternative embedded solution methods. First, both active-set (AS) and interior-point (IP) methods require rather sophisticated linear algebra routines (including matrix factorizations) for performing the underlying computations. This makes any implementation of these methods considerably complex (in terms of amount of source code) and hard to execute on devices with limited accuracy of the internal number representation (such as most PLCs or FPGAs). In order to overcome this first drawback, so-called *first-order methods* have been proposed for use in embedded optimization. They are particularly simple to implement, allow for warm-starting and can easily exploit any sparsity pattern. Unfortunately, their actual numerical performance strongly depends on the characteristics of the problem to be solved, such that these methods may be anywhere between orders of magnitudes faster to orders of magnitude slower than traditional AS or IP methods. Moreover, they share another drawback of all approaches mentioned so far: they are iterative approaches exhibiting the potential risk of slow convergence or even failure to find a solution in certain circumstances. In contrast, so-called *explicit methods* have been proposed that pre-compute the solutions to all possibly occurring QP instances offline and store them in a look-up table to be searched online. While this approach has allowed for astonishingly high sampling rates in certain applications, it suffers from the curse of dimensionality and is only applicable to embedded optimization problems featuring small problem dimensions.

In the following, we are going to briefly sketch the main ideas and existing embedded implementations of all of these four major approaches. As there exist even more methods for tackling convex QP problems, some of them are mentioned at the end of this section.

4.2 Active-set methods

Active-set methods are based on the observation that solving an equality-constrained QP problem (i.e. one without inequalities (2c)) is equivalent to solving a single linear set of equations, which is fairly trivial. Thus, AS methods start with a guess which inequality constraint will hold with equality at the optimal solution (called the *working set*) and solve the resulting equality-constrained QP problem. If it turns out that this guess was not correct, it is updated by adding inequality constraints to or removing them from the working set until the optimal solution is found. *Primal* AS methods produce a sequence of feasible iterates (i.e. ones that respect all constraints) until the optimal solution is found, while *dual* AS methods maintain dual feasibility of the iterates until one is found that is also (primal) feasible, hence optimal. MATLAB's `quadprog` function implements a primal active-set method (among others). Dual active-set methods are available in the code `qpas` (see Wills et al. (2005), based on dual algorithm proposed in Goldfarb and Idnani (1983)) and `QPSchur`, see Bartlett and Biegler (2006). A parametric active-set strategy designed for use in MPC is implemented in the open-source software `qpOASES`, see Ferreau et al. (2008, 2014).

4.3 Interior-point methods

The basic idea of IP methods is not to account for the inequality constraints (2c) explicitly, but to penalize constraint violations by means of an additional term in the objective function instead. However, in order to make this work reliably, a non-quadratic (typically logarithmic) penalty term needs to be used. The resulting equality-constrained NLP problem is then solved using Newton's method. Structure exploiting IP methods have been proposed for use in embedded optimization, e.g. by Rao et al. (1998), where the linear equation system underlying the Newton step is solved using Riccati recursions, and by Wang and Boyd (2010) whose primal IP approach employs a block-wise Cholesky factorization.

In order to increase efficiency of the implementation, code generation has become a popular approach for obtaining customized implementations of IP methods: the generic QP solver `CVXGEN` by Mattingley and Boyd (2009) exploits data sparsity by pre-computing a permutation such that the LDL^T factorization of the linear system underlying the Newton step is sparse. In contrast, the primal-dual IP solver `FORCES` presented in Domahidi et al. (2012) (later integrated into `FORCES PRO`) is tailored to the multistage structure typical of MPC problems.

Another way to increase efficiency of the implementation has been proposed in Frison et al. (2014), where the most time-critical linear algebra operations have been tailored to the intrinsic CPU features of the target hardware.

4.4 First-order methods

First-order methods may be further categorized into *splitting methods*, which are discussed in detail in Section 5, and *gradient methods*. The latter iteratively compute a step towards the solution of the unconstrained QP problem and then project this step onto the feasible set. As this projection is difficult in case of output or polytopic constraints (in fact, exact projection is as complex as solving a full QP problem), several variants to dualize some parts of the problem formulation have been proposed: the MATLAB toolbox `FiOrdOs` (see Ullmann (2011)) autogenerates code for either primal fast gradient methods (if

only simple constraints are present) or dual fast gradient methods by applying a Lagrangian relaxation to equality and polytopic constraints as proposed in Richter et al. (2011). The GPAD algorithm proposed in Patrinos and Bemporad (2012) dualizes only the inequality constraints, which allows one to either eliminate or keep the states in the primal problem. Recently, generalized versions of these fast gradient methods were proposed by Giselsson (2014) and shown to require a significantly lower number of iterations on certain problems. Also gradient methods based on inexact first-order information have been proposed (see Nedelcu et al. (2014)) and implemented in the open-source toolbox `DuQuad` as described in Kvamme (2014–2015). Finally, also the commercial embedded solver `FORCES PRO` can generate code for first-order methods (e.g. primal/dual fast gradient methods).

4.5 Explicit MPC

Bemporad et al. (2002a) pointed out to the control community that the solution of any linear MPC problem is a continuous, piece-wise affine function of the initial state $\overline{x}_0$, defined over a polyhedral partition of the feasible set. For optimization problems of sufficiently small problem dimension, this solution function can be pre-computed offline and then stored into a look-up table. Evaluating this look-up table for determining the next control input is the only computational task that remains to be done online. As this evaluation is extremely simple, it can be implemented on virtually any computing platform, as long as sufficient memory for storing the function is available. This makes explicit MPC a frequently used tool for embedded MPC implementations, particularly for systems with high sampling rates. A recent survey on explicit methods can be found in Alessio and Bemporad (2009).

The multi-parametric toolbox (`MPT`) as developed by Kvasnica et al. (2004) is a mature and popular MATLAB toolbox that implements various explicit methods such as multi-parametric quadratic programming and variants that aim at providing an only approximate solution to the QP problems at the benefit of reduced off-line or online complexity.

4.6 Further approaches

Several further approaches for solving convex QP problems have been recently proposed for use in embedded optimization. Among them are the multiplicative update dual optimization algorithm PQP by Di Cairano and Brand (2013), proposing a simple, highly parallelizable QP algorithm. In contrast, the dual Newton-type code `qpDUNES` by Frasch et al. (2014) and the piecewise-smooth Newton method proposed in Patrinos et al. (2011) are tailored to linear optimal control problems. Both make use of second-order information to converge to the optimal solution within only a few, but computationally complex iterations.

5. METHODS FOR (MORE) GENERAL CONVEX OPTIMIZATION PROBLEMS

In this section we give a brief overview of splitting methods, a wide family of conceptual schemes that decompose a (difficult) optimization problem into (significantly) simpler subproblems and solve those instead of the original one. Although the focus of splitting methods traditionally lies in large-scale optimization applications, recent advances have rendered them a competitive alternative to other optimization algorithms when it comes to embedded control. The following section is based on the extended survey Stathopoulos et al. (2016).

5.1 Splitting methods overview

We first rewrite problem (1) as a summation of two convex functions, i.e.,

$$\begin{array}{ll} \underset{w,\,v\,\in\,\mathbb{R}^n}{\text{minimize}} & h(w)+g(v) \\ \text{subject to} & w=v\,, \end{array} \tag{15}$$

where

$$h(w)=f(w)+\delta(Cw\mid\overline{c}),\quad g(w)=\delta(w\mid\Omega),$$

and $\delta(z\mid\mathcal{C})$ is an indicator function for the (convex) set $\mathcal{C}$ defined as

$$\delta(z\mid\mathcal{C})=\begin{cases}0 & z\in\mathcal{C}\\ \infty & \text{otherwise.}\end{cases}$$

A slight abuse of notation is used when the set $\mathcal{C}$ is a singleton, and $\delta(Cw\mid\overline{c})$ is written, rather than $\delta(Cw\mid\{\overline{c}\})$. Note that h and g are nonsmooth, extended real-valued functions, i.e., $f,g:\mathbb{R}^n\to\mathbb{R}\cup\{+\infty\}$.

The underlying assumption behind splitting methods is that minimizing separately over h and g is simpler (less costly) than solving (1) directly. There are a variety of ways to proceed with these minimizations, e.g., by forming (augmented) Lagrangian relaxations and then applying alternating minimization schemes, by mixing Lagrangians with augmented Lagrangians or by mixing alternating minimizations with partial linearization of the functions involved. A fairly general method from which several others can be derived is the Proximal Method of Multipliers (PMM):

$$\begin{aligned}(w^{[i+1]},v^{[i+1]})&\in\underset{w,v}{\operatorname{argmin}}\ \mathcal{P}_\rho(w,v;\lambda)\\ \lambda^{[i+1]}&=\lambda^{[i]}+\rho(w^{[i+1]}-v^{[i+1]})\ ,\end{aligned} \tag{16}$$

where $\mathcal{P}_\rho$ is the *proximal augmented Lagrangian* function defined as:

$$\begin{aligned}\mathcal{P}_\rho(w,v;\lambda)=&h(w)+g(v)+\lambda^\top(w-v)+\\ &\frac{\rho}{2}\|w-v\|^2+\frac{1}{2}\|w-w^{[i]}\|^2_{P_1}+\frac{1}{2}\|v-v^{[i]}\|^2_{P_2}.\end{aligned} \tag{17}$$

The function (17) comprises the Lagrangian function with the addition of a quadratic regularizer for $w-v$ (augmentation term) and two proximal terms ensuring that the current primal iterates w and v will not deviate much from their former values. The matrices P_1 and P_2 are positive semidefinite, while $\rho>0$ is a penalty term.

Although minimizing (17) over (w,v) simultaneously is as difficult as solving (15), it can be much easier to do it in an alternating manner. By doing so and with proper choices of the matrices P_1 and P_2, (16) gives rise to several popular splitting methods like the Alternating Direction Method of Multipliers (ADMM) and Douglas-Rachford splitting (Eckstein and Bertsekas (1992)) and the Chambolle-Pock scheme (Chambolle and Pock (2011)).

Example 1. (Douglas-Rachford splitting). Consider the case where we set $P_1 = P_2 = 0$ in (17) and minimize (16) by alternating. The iterations read

$$w^{[i+1]} = \underset{w}{\text{argmin}} \quad h(w) + (\rho/2)\|w - (v^{[i]} - \lambda^{[i]}/\rho)\|^2$$
$$v^{[i+1]} = \underset{v}{\text{argmin}} \quad \delta(v \mid \Omega) + (\rho/2)\|v - (w^{[i+1]} + \lambda^{[i]}/\rho)\|^2$$
$$\lambda^{[i+1]} = \lambda^{[i]} + \rho(w^{[i+1]} - v^{[i+1]})$$

When Douglas-Rachford splitting is applied to the dual of (15), one recovers ADMM applied to (15). Note that the method can deal with the more general case where v is an affine mapping of w, or when v and w are coupled via a linear equality of the form $Mw + Nv = b$.

5.2 Computational speedup

In linear MPC, f is often a quadratic function corresponding to some tracking or regulation objective, w is the vector of state and input variables, while $\delta(Cw \mid \bar{c})$ describes the dynamics of the system. State and input constraints can be enforced via the indicator functions $\delta(w \mid \Omega)$. This being the case, the first minimization step in Example 1, but also in several splitting schemes, amounts to solving a system of linear equations. Provided that the second step, i.e., the projection onto the constraint set Ω is often cheap, the linear system solve is the main computational burden of the algorithm. It is, therefore, worth investing some effort to speedup the solve. This can be done in several ways, e.g., by pre-factorizing the matrix involved in the inversion or by inverting it offline. When linear time invariant systems are considered the matrix does not change from one iteration to the next, and significant computational savings can be achieved. For a detailed treatment of the computational linear algebra aspect of splitting methods the interested reader is referred to (Stathopoulos et al., 2016, Chapter 5).

5.3 Bells and whistles

In the absence of second order information (Hessians of the smooth terms), splitting methods share many of the weaknesses of gradient-based optimization algorithms. These issues can be mitigated by using several heuristic and non-heuristic approaches that 'polish' the methods and improve their performance.

Conditioning. As is the case with every first order method, splitting schemes suffer from slow convergence when the problem data are ill-conditioned. Preconditioning the data offline is often very helpful to recover performance. Roughly speaking, by scaling the problem data we choose a new coordinate system that adjusts better to the geometry of the problem. The adjustment typically takes place in the dual space, i.e., the level sets of the smooth part of the dual form of (15) are shaped to become 'more uniform'. More details on preconditioning of splitting methods are given in Giselsson and Boyd (2015, 2017).

Acceleration. Inspired by Nesterov's accelerated gradient (Nesterov (1983)) and Tseng's generalization to proximal gradient methods (Tseng (2008)), several splitting schemes have been accelerated by means of adaptive relaxation sequences. The performance of the accelerated variants is often remarkable, especially given that the extra cost per iteration is almost negligible. Refinements of accelerated methods have been developed in O'Donoghue and Candes (2015), where the acceleration is occasionally halted so as to avoid overly aggressive steps that cause oscillatory behavior.

Warm-starting. Linear MPC requires the solution of a sequence of convex optimization problems which do not differ significantly from each other. The change often concerns the affine part of the objective function (e.g., tracking a time varying signal) and the right-hand side of the linear equality constraints (dynamics), where the updated state appears (see (4)). Splitting methods can be easily warm-started by initializing the new optimal control problem to the optimal primal and dual iterates of the previous one. Assuming that the problems are reasonably similar, this heuristic improves the convergence performance (see (Stathopoulos et al., 2016, Chapter 6) for several examples).

Non-linearity. Recent results (Attouch and Bolte (2009); Attouch et al. (2013)) have provided a very general framework for proving global convergence of descent methods on non-smooth semi-algebraic objectives. These techniques can be exploited to develop splitting, or decomposition algorithms for nonlinear problems that require similar operators and provide similar benefits to the more classic convex approaches (Hours and Jones (2016)). When deploying an embedded nonlinear MPC solution, it is often the case that only a very small number of optimization steps can be completed in each sample period, before the input is applied to the system, the state estimate is updated, and the parametric problem to be solved changes. The resulting suboptimality error for these continuously changing family of problems has been studied, and general conditions under which this error is stable have been developed (Hours and Jones (2016)).

5.4 Summary

We conclude this section by giving a rough guideline on when splitting methods should be used. Two important aspects to keep in mind are that splitting schemes are only useful if (i) the original problem is decomposable into subproblems that are much easier to solve (ideally they have closed form solutions) and (ii) when a solution of low to medium accuracy is sufficient. Splitting methods can tackle a wide variety of convex objectives and constraints, a characteristic that makes them more general than quadratic programming solvers. In addition, they often enjoy a low memory footprint and are very simple to code, requiring a few tens of lines of code consisting of simple linear algebra operations. The latter is an appealing characteristic for embedded control.

6. METHODS FOR NONLINEAR OPTIMAL CONTROL PROBLEMS

The focus of this section lies on discussing various algorithms for solving nonlinear optimal control problems, tailored to real-time and embedded applications. We will be specific on real-time methods for MPC. The methods that we mention in this section are also perfectly suited for MHE, due to the stark resemblance of MPC and MHE, however we will stick to the "control" point of view, for sake of simplicity.

6.1 Newton-type methods

The overarching strategy of the methods discussed in the following is Newton's method. This method yields iterates $w^{[i]}, i = 0, 1, \ldots$ that ultimately return a solution of a set of nonlinear equations $R(w) = 0$. In the context of optimal control, we solve nonlinear OCPs as defined in (8) by finding approximations $(w^{[i]}, \lambda^{[i]}, \mu^{[i]}), i = 0, 1, \ldots,$ for a solution to the KKT conditions (7).

Two major families of Newton-type algorithms to solve nonlinear OCPs exist. They differ in their treatment of the non-smooth complementarity conditions (7c)-(7e). This leads to the two classes of *sequential quadratic programming (SQP)* and *nonlinear interior-point (NIP)* methods, respectively, which are sketched in the following.

6.2 SQP methods

A first way of iteratively solving the KKT system (7) is to linearize its nonlinear equations (7a)-(7d) at the current iterate $(w^{[i]}, \lambda^{[i]}, \mu^{[i]})$, and from there compute a new approximation by solving a QP problem in each iteration. More specifically, the resulting linearized complementarity system corresponds to the KKT conditions of the following QP:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & \frac{1}{2}(w - w^{[i]})^\top H (w - w^{[i]}) \\ & + g^\top (w - w^{[i]}) \\ \text{subject to} \quad & c_{\text{lin}}(w) = 0, \\ & d_{\text{lin}}(w) \leq 0\,, \end{aligned} \tag{18}$$

of which the structure corresponds to the one of NLP (8), i.e.

$$\begin{aligned} w^{[i]} &= \left[x_0^{[i]}, u_0^{[i]}, x_1^{[i]}, \ldots, x_N^{[i]}\right]^\top, \qquad \text{as before,} \\ H &= \nabla_w^2 \mathcal{L}(w^{[i]}, \lambda^{[i]}, \mu^{[i]}), \qquad \text{as in (9),} \\ g &= \nabla_w f(w^{[i]}), \\ c_{\text{lin}}(w) &= \begin{bmatrix} x_0 - \overline{x}_0 \\ x_1 - x_1^{[i]} - F_0(x_0^{[i]}, u_0^{[i]}) - [A_0 \;\; B_0] \begin{bmatrix} x_0 - x_0^{[i]} \\ u_0 - u_0^{[i]} \end{bmatrix} \\ \vdots \end{bmatrix} \\ d_{\text{lin}}(w) &= \begin{bmatrix} d(x_0^{[i]}, u_0^{[i]}) + [D_0^x \;\; D_0^u] \begin{bmatrix} x_0 - x_0^{[i]} \\ u_0 - u_0^{[i]} \end{bmatrix} \\ \vdots \\ d(x_N^{[i]}) + D_N^x (x_N - x_N^{[i]}) \end{bmatrix}, \end{aligned}$$

with the definitions of $x_k, u_k, \overline{x}_0$ as in (4), as well as D_k^x, D_k^u, which now form the linearization of the nonlinear inequality constraints at the current iterate $w^{[i]}$. Furthermore, we linearized the discrete dynamic equations (8b) at $w^{[i]}$, resulting in the system matrices $A_k \in \mathbb{R}^{n_x \times n_x}$ and $B_k \in \mathbb{R}^{n_x \times n_u}$. Note that the structure of (18) is the same as that of linear MPC problem (4), with the only difference that all matrices are varying over the MPC horizon. For more details about the connection between linear and nonlinear MPC we refer the reader to Gros et al. (2017).

The primal and dual solution of QP (18) form the updated iterates $(w^{[i+1]}, \lambda^{[i+1]}, \mu^{[i+1]})$. It can be solved directly by using a structure-exploiting QP solver, e.g. `HPMPC` (Frison et al. (2014)), `qpDUNES` (Frasch et al. (2015)), `FORCES` (Domahidi et al. (2012)). Alternatively, we can use the so-called *condensing* approach (Bock and Plitt (1984)), which results in a much smaller QP due to the elimination of the dynamic constraints, namely the equality constraints in (18). Any general purpose dense QP solver can then be applied to solve the smaller QP. Worth noting in this context is the QP solver `qpOASES` (Ferreau et al. (2014)), which exploits the fact that subsequent problems in real-time MPC lie close to each other, as we will explain in Section 6.4.

The exact Hessian $\nabla_w^2 \mathcal{L}(w^{[i]}, \lambda^{[i]}, \mu^{[i]})$ might become indefinite at some iterates. In this case, the step $(w^{[i+1]} - w^{[i]})$ in (18) is not guaranteed to be a descent direction, which is highly unfavorable. One remedy is to apply *regularization*. This consists of adding curvature to the original Hessian in order to obtain a positive definite approximation.

Apart from the above exact-Hessian based approach, many widely-used variants of the SQP algorithm exist that use an approximation to the Hessian. We mention one which is of particular interest to embedded application of receding horizon control.

The Generalized Gauss-Newton method One example of an SQP variant that has been proven to work well in practice is the Generalized Gauss-Newton (GGN) method (Bock (1983)). This method applies to least-squares problems, e.g. (10). In solving such types of problems, instead of using the exact Hessian as explained in Section (6.2), we can use the following Hessian approximation which is cheaper to compute:

$$H_{\text{GGN}} = \frac{\partial h(w^{[i]})}{\partial w}^\top \frac{\partial h(w^{[i]})}{\partial w}, \tag{19}$$

where we stack the nonlinear output functions of (10) in $h = [h_0^\top, \ldots, h_N^\top]$. Note that we do not compute second-order sensitivities, and that the Gauss-Newton Hessian is positive semi-definite by construction. Furthermore, no Lagrange multipliers are required for its computation.

A special case occurs if the nonlinear OCP objective is a convex quadratic ($f(w) = \frac{1}{2} w^\top H_{\text{OCP}}\, w + g_{\text{OCP}}^\top w$, with $H_{\text{OCP}} \succ 0$), then GGN consists of performing SQP with subproblems (18), but with $H = H_{\text{OCP}}$.

When is the Gauss-Newton Hessian a good approximation of the exact Hessian? In Verschueren et al. (2016), it is proven for an SQP method with full steps that a local

minimizer of the nonlinear problem becomes unattractive if the approximate Hessian is smaller (in the matrix sense) than half of the exact Hessian. For least-squares problems as (10), this problem is avoided if the problem is only mildly nonlinear or when the residuals are small.

One generalization of the GGN algorithm worth mentioning is Sequential Convex Programming (SCP, Tran-Dinh et al. (2012)), applicable to NMPC formulations with more general convex objective function or inequality constraints. SCP still does linearize the non-convex dynamic equations, but passes off the convex objective and/or inequalities to a general-purpose convex solver.

6.3 Nonlinear interior-point methods

In interior point methods, the way of treating the nonlinear KKT system (7) is to smoothen the complementarity conditions (7c)-(7e) by introducing the so-called *barrier parameter* $\tau > 0$, such that the perturbed KKT conditions read as

$$\nabla_w \mathcal{L}(w, \lambda, \mu) = 0, \tag{20a}$$
$$c(w) = 0, \tag{20b}$$
$$\mu_i d_i(w) + \tau = 0, \quad i = 1, \ldots, n_{\text{iq}}. \tag{20c}$$

System (20) is then solved with Newton's method, for a given value of the barrier parameter τ. Afterwards, τ is adapted appropriately and the next iteration starts. The structure of the linear systems that need to be solved in each step is identical to the ones in interior point methods for convex quadratic problems. The difference is that the quantities are linearized in each iteration. Note that, similar as in SQP methods, one can use an approximation of the exact Hessian $\nabla^2_w \mathcal{L}(w, \lambda, \mu)$ for the Newton iterations instead.

An important consideration in nonlinear interior point methods is the strategy for updating the barrier parameter. Different strategies exist, and a good strategy for interior point methods for linear or quadratic programs does not necessarily perform well on nonlinear problems (Nocedal et al. (2009)).

Another algorithmic component is globalization, in order to ensure convergence to a local optimum. Recently, filter line-search methods became popular (Wächter and Biegler (2006a)). An excellent implementation is the package `IPOPT` (Wächter and Biegler (2006b)). Other codes that are more practical for use on embedded and/or real-time platforms, are `FORCES PRO` (Zanelli et al. (2016)) and `PIPS-NLP` (Chiang et al. (2016)).

6.4 Online methods for nonlinear MPC

A useful realization in the context of receding horizon control is that the subsequent optimization problems that we are trying to solve bear a striking similarity. Often, the state of the underlying plant does not change all that much, depending on the difference between the time constant of the plant and the sampling time of the controller.

Supposing that we have a solution to some instant of the online control problem, we can do at least two things to prepare for the control problem in the next time instant: 1) we use an initial guess which consists of the solution of the previous problem shifted in time, 2) we employ a so-called *tangential predictor* that gives us information of the optimization problem (both IP and SQP methods deliver such a tangential predictor, but unfortunately the interior point solution manifold becomes highly nonlinear at an active set change, which results in a less accurate predictor for those methods compared to SQP). By combining shifting and the use of a tangential predictor, we usually end up with an adequate initial guess for the next optimal control problem instance.

However, when applying an advanced optimal control method in an online fashion, like NMPC, we face the following problem. At each time instant, we estimate the current state, upon which we base our optimal control problem. Solving this problem takes time, which makes our current state estimate quickly become *outdated*. This might be called the real-time dilemma (Diehl (2001)).

We can take either of two options to solve this dilemma. The first option is to compensate for the computational delay introduced by the method at hand. This approach is taken in e.g. the *advanced step controller* by Zavala and Biegler (2009), which is based on a nonlinear interior point algorithm. It takes into account a prediction of what the initial state will be when the computations will be ready.

The second option is to not fully solve the optimal control problem to convergence. This is opted for in e.g. the Newton-type controller (Li and Biegler (1989)), which only performs one full Newton SQP-step per sampling time. This method is based on a *single shooting* discretization and does not make use of a tangential predictor. The Continuation/GMRES method by Ohtsuka (2004) also takes just one Newton step, but is based on an interior point algorithm, and by contrast does make use of a tangential predictor.

An efficient implementation of the Continuation/GMRES method is the `AutoGenU` package (Ohtsuka and Kodama (2002)) and an example of experimental validation of this method is done on a radio-controlled hovercraft (Seguchi and Ohtsuka (2003a)).

Another approach for online NMPC and NMHE that has been proven to be efficient in practice is the real-time iteration (RTI) scheme (Diehl et al. (2002)). This SQP-type method only solves one QP (18) per sampling instant. The underlying idea is to approach a local solution *while* controlling the plant. The RTI scheme has some attractive properties for practical application. First, it makes use of a *generalized* tangential predictor, which can be shown to work well across active set changes, even when started from an approximate solution (we refer to Diehl et al. (2009) for a more in-depth comparison between the different types of tangential predictors). Secondly, we can save time by splitting our computations in a *feedback* and *preparation* phase. The preparation phase consists of simulation of the nonlinear system, as well as generating first- and second-order derivative information. Also condensing is part of this phase. The feedback phase starts when the measurement of the initial state becomes available, and consists just of the solution of one QP, thus minimizing feedback delay.

A popular implementation of the RTI is the `ACADO Toolkit` (Houska et al. (2011)). Special code generated integrators for embedded optimization exploit the iterative nature of the process and allow for a significant CPU time reduction (Quirynen et al., 2017). Many examples of experimental application of the RTI scheme exist, e.g. on electrical drives in the megawatt range (Besselmann et al. (2016b)), on small-scale race cars (Verschueren et al. (2014)), on autonomous agricultural vehicles (Kayacan et al. (2014)) and on a 6-DOF robotic manipulator (van Duijkeren et al. (2016)).

7. METHODS FOR OPTIMAL CONTROL PROBLEMS WITH DISCRETE DECISIONS

Receding horizon optimal control problems involving discrete decisions can be broadly classified into pure- and mixed-integer problems. Pure-integer (or integer) problems are optimization problems, where all decision variables are restricted to a subset of the integers, i.e., all decision variables are discrete. Such problems are often related to scheduling and planning, many of which have been studied extensively and have counterparts as optimization over graphs, such as the caterer problem (Jacobs (1954)), the vehicle routing problem (Dantzig and Ramser (1959)), or the assignment problem (Martello and Toth (1987)). For these problems, tailored methods for finding global or approximate solutions exist, some of which can be implemented in an embedded context.

In most cases, however, control problems involve both discrete and continuous quantities and therefore fall in the latter category of mixed-integer problems. In the following, we will elaborate on different solution methods for mixed-integer problems, that are suited for embedded control applications.

7.1 Explicit solutions

Similar to the linear MPC case as discussed in Section 4.5, the solution of mixed-integer MPC problems of form (14) is likewise a piece-wise affine function of the initial state, $\overline{x}_0$ (Bemporad et al. (2002b)). However, it is usually discontinuous and its domain is non-convex, i.e., a finite union of polyhedra. Furthermore, the number of regions of the piece-wise affine function grows exponentially, not only in the control horizon N, but also in the number of binary optimization variables. Generating and storing the explicit solution therefore becomes a major limitation, and is feasible only for problems with few binary variables and a short horizon N. If the explicit solution can be generated, it can be evaluated very efficiently using a binary tree search, see Tøndel et al. (2003); Christophersen et al. (2007). Furthermore, it is possible to obtain suboptimal explicit solutions with reduced storage complexity, see Axehill et al. (2014).

Generating explicit solutions of mixed-integer MPC problems, as well as efficient embeddable C-code, is implemented as part of the multi-parametric toolbox (Herceg et al. (2013)) for MATLAB. Also the hybrid toolbox (Bemporad (2004)) for MATLAB implements tools for the generation of explicit MPC controllers involving discrete decisions.

7.2 Enumeration-based methods

Most effective online methods for solving mixed-integer problems rely on some form of enumeration of the discrete decisions. Methods based on pure enumeration are sometimes used (Geyer (2005); Rodriguez et al. (2013)), particularly for input-switched/switched-affine systems with very short control horizons. The trajectory of such systems is fully determined by the (measured) initial state and the sequence of switching inputs. Therefore, the trajectories resulting from all possible switching sequences can be predicted numerically and the sequence leading to the lowest-cost trajectory chosen. This technique can be used for arbitrary (possibly nonconvex) cost, constraints and discrete-time dynamics. It can even be applied to continuous-time dynamics, using numerical integrators. The required computation grows exponentially in control horizon N, but is fixed and known a-priori, making this method well-suited for embedded applications. For systems with continuous inputs or larger control horizon, this method becomes impractical, and smarter methods of enumeration are required.

The most popular of these techniques is *branch-and-bound* (Wolsey and Nemhauser (1988); Bertsimas and Weismantel (2005)). For problems, where the only source of nonconvexity is the discrete nature of some of the decision variables, e.g. in a convex optimization problem of form (1), where some of the decision variables are restricted to be integers, a hierarchy (or tree) of relaxations can be established. Each node in the tree corresponds to fixing one of the discrete decision variables to a particular value and relaxing all others to an interval containing all possible discrete decisions of that variable. The branch-and-bound algorithm traverses this tree, solving the relaxation, a convex optimization problem, at each node. The solution of this relaxed problem provides a (local) lower bound on the objective value for all nodes further down in the tree, i.e., nodes where more discrete decision variables are fixed. The decision of determining which node of the tree to explore next is called *branching*. Traversing the tree downwards, fixing more and more discrete variables, leads to better and better lower bounds. Leaf nodes of the tree represent convex optimization problems, where all discrete decisions have been fixed. If the convex problem at a leaf node is feasible, we obtain a feasible solution of the original mixed-integer problem, and therefore a global upper bound on the optimal objective value. If at some node, the local lower bound exceeds the best global upper bound, none of the nodes further down the tree need to be explored, since they can only lead to suboptimal solutions. This is called *bounding* and helps avoiding to enumerate all possible combinations of discrete decisions. The method of *branch-and-cut* combines branching with cutting-plane techniques (Wolsey and Nemhauser, 1988). In branch-and-cut, the relaxed problems are augmented with constraints that are redundant for the original mixed-integer problem but reduce the feasible set of the relaxed problem, reducing the number of nodes that have to be explored. Both of these methods can be extended to be used for solving general MINLP problems, using *spatial* branch-and-bound, with tools such as `BARON` (Sahinidis (2014)), `Bonmin` (Bonami et al. (2013)) or `Couenne` (Belotti et al. (2009)).

Solving general MINLPs is very rarely feasible in an embedded context. However, branch-and-bound is particularly well suited for solving MIQPs (Fletcher and Leyffer (1998)), in part, because the relaxed problems, needed to be solved at each node, are simple, convex quadratic programs. In control, mixed-integer MPC problems can be formulated as MIQPs, see (14), where the multistage structure of the optimization problem can be exploited to further improve the speed at which the relaxed QPs can be solved, thereby improving the overall computational efficiency of the branch-and-bound scheme. This fact has been exploited in (Axehill, 2008, p. 123ff), where a dual active-set method tailored to the MPC structure was used as a solver for the relaxed problems in branch-and-bound. In Frick et al. (2015) a method for generating embedded C-code for a parametric branch-and-bound solver that can be tailored to the particular mixed-integer MPC problem was presented. The method utilizes `FORCES` (Domahidi et al. (2012)), a tool for generating customized interior-point solvers for parametric QPs. `FORCES PRO` (Domahidi and Jerez (2014)) can now generate tailored statically allocated branch-and-bound solvers. Similarly, in Bemporad (2015), a tailored active-set method based on non-negative least squares is proposed as a relaxed QP solver in combination with branch-and-bound. The active-set method features warm-starting and early-stopping, which can be used to accelerate the branch-and-bound procedure, particularly in a receding horizon setting. Heuristics for improving the performance of branch-and-bound schemes, tailored to mixed-integer MPC problems, are discussed in Frick et al. (2015). In Axehill et al. (2010), it is shown that semi-definite programming (SDP) relaxations are at least as tight as QP relaxations, and in Moehle and Boyd (2015) a perspective-based formulation for switched-affine systems, with general convex programs as relaxations, is presented. These relaxations are also at least as tight as the QP relaxations. Such methods can lead to fewer nodes that have to be explored in a branch-and-bound scheme, however efficient SDP or general convex solvers are not yet available in an embedded context.

7.3 Splitting methods

With the recent interest in methods based on operator splitting for solving convex and smooth nonconvex problems, there have been efforts to develop similar methods for nonsmooth problems arising in MPC with discrete decisions. As in the convex case, a consensus problem of form (15) is considered, where the problem is split into two parts h and g. In the case of mixed-integer problems the function $g(w) = \delta(w \mid \Omega)$ is the indicator function for a nonconvex set and contains constraints modeling the discrete nature of the problem. Ω is usually chosen such that g becomes separable for each stage of the MPC problem, i.e, $\Omega = \Omega_0 \times \ldots \times \Omega_N$. Douglas-Rachford splitting/ADMM (see Example 1 in Section ref) is used as a heuristic in Takapoui et al. (2016) to solve problems of this type, when a solution w to the projection $\pi_\Omega(\overline{w}) := \operatorname{argmin}_{w\in\Omega} \|w - \overline{w}\|_2$ onto the set Ω can be computed efficiently. Neither optimality, nor convergence guarantees are available in this case, which is a topic of active research. In Frick et al. (2016) a method based on operator splitting was developed to find KKT points of (15). It utilizes the PWA description of the hybrid dynamics, yielding a nonconvex set of the form $\Omega = \bigtimes_{k=0}^{N} \bigcup_{i=1}^{M} \Omega_{k,i}$, where the sets $\Omega_{k,i}$ are simple, low-dimensional polyhedra. Evaluating the projection $\pi_\Omega(\overline{w})$ reduces to projections onto the individual sets $\Omega_{k,i}$ and then choosing the minimum distance solution among the solutions of $\pi_{\Omega_{k,i}}(\overline{w}_k)$ for each k. For some cases, guarantees for local convergence and local optimality are given. Both of these methods are very simple, can be warmstarted and are therefore well suited to be implemented in an embedded setting. Furthermore, they can be used to handle larger problems, for which using explicit solutions is not feasible. Finally, they often outperform solvers based on branch-and-bound.

7.4 Other approaches

Other notable approaches include Hempel (2015), where using inverse optimization, hybrid MPC problems with piece-wise affine dynamics are transformed into linear complementarity problems with special structure. These linear complementarity problems can be solved using standard NLP solvers, and often outperform commercial mixed-integer programming solvers.

In the case of integer controls and nonlinear continuous dynamics, a convexification strategy was developed by Sager (2005) that allows one to efficiently obtain solutions of a relaxed problem that can be transformed into ϵ-feasible solution approximations whose error can be bounded in practice and in theory (Sager et al., 2012).

8. CONCLUSIONS

Embedded optimization is becoming a mature technology and the field is still developing and expanding quickly. While most applications used to rely on rather simple, convex QP formulations just a decade ago, nowadays even highly nonlinear or mixed-integer formulations for industrially relevant problems are solvable within a couple of milliseconds on embedded computing hardware. This survey provided an overview of the most promising algorithmic concepts as well as state-of-the-art implementations.

Current trends such as "internet of things" are about to make embedded computing resources ubiquitous, naturally asking for smart software to harvest their full potential. For many applications it will not be sufficient to just control systems and devices decently. Rather, it is going to be expected that they behave optimally and in an autonomous manner. Embedded optimization will not be the answer to all questions arising from this evolution, but it is likely to be a key building block.

REFERENCES

Abdollahpouri, M., Takács, G., and Rohal'-Ilkiv, B. (2017). Real-time moving horizon estimation for a vibrating active cantilever. *Mechanical Systems and Signal Processing*, 86, 1–15.

Alessio, A. and Bemporad, A. (2009). *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, chapter A Survey on Explicit Model Predictive Control, 345–369. Springer.

Almér, S., Mariéthoz, S., and Morari, M. (2013). Sampled Data Model Predictive Control of a Voltage Source Inverter for Reduced Harmonic Distortion. *IEEE Transactions on Control Systems Technology*, 21(5), 1907–1915.

Almér, S., Mariéthoz, S., and Morari, M. (2015). Dynamic Phasor Model Predictive Control of Switched Mode Power Converters. *IEEE Transactions on Control Systems Technology*, 23(1), 349–356.

Attouch, H. and Bolte, J. (2009). On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116, 5–16.

Attouch, H., Bolte, J., and Svaiter, B. (2013). Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting and regularised Gauss-Seidel methods. *Mathematical Programming*, 137, 91–129.

Axehill, D. (2008). *Integer Quadratic Programming for Control and Communication*. Ph.D. thesis, Linköping University, The Institute of Technology.

Axehill, D., Besselmann, T., Raimondo, D.M., and Morari, M. (2014). A parametric branch and bound approach to suboptimal explicit hybrid MPC. *Automatica*, 50(1), 240–246.

Axehill, D., Vandenberghe, L., and Hansson, A. (2010). Convex relaxations for mixed integer predictive control. *Automatica*, 46(9), 1540–1545.

Bartlett, R. and Biegler, L. (2006). QPSchur: A Dual, Active Set, Schur Complement Method for Large-scale and Structured Convex Quadratic Programming Algorithm. *Optimization and Engineering*, 7, 5–32.

Beccuti, A.G., Mariethoz, S., Cliquennois, S., Wang, S., and Morari, M. (2009). Explicit model predictive control of DC-DC switched-mode power supplies with extended kalman filtering. *IEEE Transactions on Industrial Electronics*, 56(6), 1864–1874.

Belotti, P., Lee, J., Liberti, L., Margot, F., and Wächter, A. (2009). Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4-5), 597–634.

Bemporad, A. (2004). Hybrid toolbox - user's guide. `http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox`.

Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407–427.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002a). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38, 3–20.

Bemporad, A. (2015). Solving mixed-integer quadratic programs via nonnegative least squares. *IFAC-PapersOnLine*, 48(23), 73–79.

Bemporad, A., Borrelli, F., and Morari, M. (2002b). *On the Optimal Control Law for Linear Discrete Time Hybrid Systems*, 105–119. Springer, Berlin Heidelberg.

Bertsimas, D. and Weismantel, R. (2005). *Optimization over integers*, volume 13. Dynamic Ideas Belmont.

Besselmann, T.J., Almér, S., and Ferreau, H.J. (2016a). Model Predictive Control of Load-Commutated Inverter-Fed Synchronous Machines. *IEEE Transactions on Power Electronics*, 31(10), 7384–7393.

Besselmann, T.J., de moortel, S.V., Almér, S., Jörg, P., and Ferreau, H.J. (2016b). Model predictive control in the multi-megawatt range. *IEEE Transactions on Industrial Electronics*, 63(7), 4641–4648.

Biegler, L. (1984). Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8, 243–248.

Biegler, L. and Zavala, V. (2009). Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization. *Computers and Chemical Engineering*, 33(3), 575–582.

Bock, H.G. (1983). Recent advances in parameter identification techniques for ODE. In *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, 95–121. Birkhäuser.

Bock, H.G. and Plitt, K.J. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the IFAC World Congress*, 242–247. Pergamon Press.

Boechat, M.A., Liu, J., Peyrl, H., Zanarini, A., and Besselmann, T. (2013). An architecture for solving quadratic programs with the fast gradient method on a field programmable gate array. In *Proceedings of the 21st Mediterranean Conference on Control Automation (MED)*, 1557–1562.

Bonami, P., Belotti, P., D'Ambrosio, C., Forrest, J.J., Ladanyi, L., Laird, C., Lee, J., Margot, F., Vigerske, S., and Waechter, A. (2013). Bonmin (basic open-source mixed integer programming). URL `https://projects.coin-or.org/Bonmin`.

Camacho, E. and Bordons, C. (2007). *Model Predictive Control*. Springer, 2nd edition.

Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1), 120–145.

Chiang, N.Y., Hang, R., and Zavala, V.M. (2016). An augmented lagrangian filter method for real-time embedded optimization. URL `http://www.optimization-online.org/DB_HTML/2016/10/5664.html`.

Christophersen, F.J., Kvasnica, M., Jones, C.N., and Morari, M. (2007). Efficient evaluation of piecewise control laws defined over a large number of polyhedra. In *European Control Conference*, 2360–2367.

Dantzig, G.B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Dantzig, G.B. and Ramser, J.H. (1959). The truck dispatching problem. *Management Science*, 6(1), 80–91.

Darby, M. and Nikolaou, M. (2012). MPC: Current practice and challenges. *Control Engineering Practice*, 20, 328–342.

Debrouwere, F., Vukov, M., Quirynen, R., Diehl, M., and Swevers, J. (2014). Experimental validation of combined nonlinear optimal control and estimation of an overhead crane. *IFAC Proceedings Volumes*, 47(3), 9617–9622.

Defraene, B., van Waterschoot, T., Ferreau, H., Diehl, M., and Moonen, M. (2012). Real-time perception-based clipping of audio signals using convex optimization. *Transactions on Audio, Speech and Language Processing*, 20(10), 2657–2671.

Di Cairano, S. and Brand, M. (2013). On a multiplicative update dual optimization algorithm for constrained linear MPC. In *Proceedings of the 52nd IEEE Conf. on Decision and Control, Firenze, Italy*, 1696–1701.

Di Cairano, S., Liang, W., Kolmanovsky, I.V., Kuang, M.L., and Phillips, A.M. (2013). Power smoothing energy management and its application to a series hybrid powertrain. *IEEE Transactions on control systems technology*, 21(6), 2091–2103.

Di Cairano, S., Yanakiev, D., Bemporad, A., Kolmanovsky, I.V., and Hrovat, D. (2012). Model predictive idle speed control: Design, analysis, and experimental evaluation. *IEEE Transactions on Control Systems Technology*, 20(1), 84–97.

Diehl, M. (2001). *Real-Time Optimization for Large Scale Nonlinear Processes*. Ph.D. thesis, Universität Heidelberg.

Diehl, M., Bock, H.G., Schlöder, J., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585.

Diehl, M., Ferreau, H.J., and Haverbeke, N. (2009). Efficient numerical methods for nonlinear mpc and moving horizon estimation. In L. Magni, M. Raimondo, and F. Allgöwer (eds.), *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, 391–417. Springer.

Domahidi, A., Zgraggen, A., Zeilinger, M., Morari, M., and 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)*, 668 – 674. Maui, HI, USA.

Domahidi, A. and Jerez, J. (2014). FORCES professional. embotech GmbH (`http://embotech.com/FORCES-Pro`).

Eckstein, J. and Bertsekas, D.P. (1992). On the Douglas-Rachford Splitting Method and the Proximal Point Algorithm for Maximal Monotone Operators. *Mathematical Programming*, 55(1), 293–318.

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

Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

Ferreau, H., Ortner, P., Langthaler, P., del Re, L., and Diehl, M. (2007). Predictive control of a real-world diesel engine using an extended online active set strategy. *Annual Reviews in Control*, 31(2), 293–301.

Fletcher, R. and Leyffer, S. (1998). Numerical experience with lower bounds for MIQP branch-and-bound. *Optimization, SIAM Journal on*, 8(2), 604–616.

Frasch, J.V., Sager, S., and Diehl, M. (2015). A parallel quadratic programming method for dynamic optimization problems. *Mathematical Programming Computations*, 7(3), 289–329.

Frasch, J.V., Vukov, M., Ferreau, H.J., and Diehl, M. (2014). A new quadratic programming strategy for efficient sparsity exploitation in SQP-based nonlinear MPC and MHE. In *Proceedings of the 19th IFAC World Congress*.

Frick, D., Domahidi, A., and Morari, M. (2015). Embedded optimization for mixed logical dynamical systems. *Computers & Chemical Engineering*, 72, 21–33. A Tribute to Ignacio E. Grossmann.

Frick, D., Domahidi, A., Vukov, M., Mariethoz, S., Diehl, M., and Morari, M. (2012). Moving horizon estimation for induction motors. In *Sensorless Control for Electrical Drives (SLED), 2012 IEEE Symposium on*.

Frick, D., Jerez, J.L., Domahidi, A., Georghiou, A., and Morari, M. (2016). Low-complexity iterative method for hybrid MPC. *arXiv preprint:1609.02819*, 1–27. URL `http://arxiv.org/abs/1609.02819`.

Frison, G., Sorensen, H.B., Dammann, B., and Jørgensen, J.B. (2014). High-performance small-scale solvers for linear model predictive control. In *Proceedings of the European Control Conference (ECC)*, 128–133.

García, C., Prett, D., and Morari, M. (1989). Model Predictive Control: Theory and Practice – a Survey. *Automatica*, 25, 335–348.

Geyer, T., Papafotiou, G., Frasca, R., and Morari, M. (2008). Constrained optimal control of the step-down dc-dc converter. *Power Electronics, IEEE Transactions on*, 23(5), 2454–2464.

Geyer, T. (2005). *Low complexity model predictive control in power electronics and power systems*. Ph.D. thesis, Eidgenössische Technische Hochschule ETH Zürich, Nr. 15953, 2005.

Giselsson, P. (2014). Improved fast dual gradient methods for embedded model predictive control. In *Proceedings of the 19th IFAC World Congress, Cape Town, South Africa*, 2303–2309.

Giselsson, P. and Boyd, S. (2015). Metric Selection in Fast Dual Forward-Backward Splitting. *Automatica*, 62, 1–10.

Giselsson, P. and Boyd, S. (2017). Linear convergence and metric selection in douglas-rachford splitting and ADMM. *To appear in IEEE Transactions on Automatic Control*, 62(2), 532–544.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27, 1–33.

Gros, S., Zanon, M., Quirynen, R., Bemporad, A., and Diehl, M. (2017). From linear to nonlinear MPC: bridging the gap via the real-time iteration. *International Journal of Control*. (in press).

Harjunkoski, I., Nyström, R., and Horch, A. (2009). Integration of scheduling and control – Theory or practice? *Computers & Chemical Engineering*, 33(12), 1909–1918. FOCAPO 2008 – Selected Papers from the Fifth International Conference on Foundations of Computer-Aided Process Operations.

Hartley, E. and Maciejowski, J. (2010). Predictive control for spacecraft rendezvous in an elliptical orbit using an FPGA. In *Proceedings of the European Control Conference*, 1359–1364. Zurich, Switzerland.

Heemels, W., Schutter, B.D., and Bemporad, A. (2001). Equivalence of hybrid dynamical models. *Automatica*, 37(7), 1085–1091.

Hempel, A.B. (2015). *Control of Piecewise Affine Systems Through Inverse Optimization*. Ph.D. thesis, ETH Zürich. Dissertation, ETH-Zürich, Nr. 23222, 2015.

Herceg, M., Kvasnica, M., Jones, C.N., and Morari, M. (2013). Multi-parametric toolbox 3.0. In *Proceedings of the European Control Conference*, 502–510. Zürich, Switzerland.

Hours, J.H. and Jones, C.N. (2016). A parametric nonconvex decomposition algorithm for real-time and dis-

tributed nmpc. *IEEE Transactions on Automatic Control*, 61(2), 287–302.

Houska, B., Ferreau, H.J., and Diehl, M. (2011). An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10), 2279–2285.

Huyck, B., Ferreau, H., Diehl, M., Brabanter, J.D., Impe, J.V., Moor, B.D., and Logist, F. (2012). Towards online model predictive control on a programmable logic controller: Practical considerations. Article ID 912603.

Iles, S., Matusko, J., and Kolonic, F. (2014). Real-time Predictive Control of 3D tower crane. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, 224–230.

Jacobs, W. (1954). The caterer problem. *Naval Research Logistics Quarterly*, 1(2), 154–165.

Jerez, J., Goulart, P., Richter, S., Constantinides, G., Kerrigan, E., and Morari, M. (2014). Embedded Online Optimization for Model Predictive Control at Megahertz Rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica*, 4, 373–395.

Kayacan, E., Kayacan, E., Ramon, H., and Saeys, W. (2014). Distributed nonlinear model predictive control of an autonomous tractor-trailer system. *Mechatronics*, 24(8), 926–933.

Kraus, T., Ferreau, H.J., Kayacan, E., Ramon, H., De Baerdemaeker, J., Diehl, M., and Saeys, W. (2013). Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. *Computers and electronics in agriculture*, 98, 25–33.

Kvamme, S. (2014–2015). DuQuad Webpage. `http://sverrkva.github.io/duquad/`.

Kvasnica, M., Grieder, P., Baoti, M., and Morari, M. (2004). Multi parametric toolbox (MPT). In R. Alur and G. Pappas (eds.), *HSCC (Hybrid Systems: Computation and Control)*, volume 2993 of *Lecture Notes in Computer Science*, 448–462. Springer Verlag.

Li, W. and Biegler, L. (1989). Multistep, Newton-Type Control Strategies for Constrained Nonlinear Processes. *Chem. Eng. Res. Des.*, 67, 562–577.

Liniger, A., Domahidi, A., and Morari, M. (2015). Optimization-based autonomous racing of 1: 43 scale RC cars. *Optimal Control Applications and Methods*, 36(5), 628–647.

Mariethoz, S. and Morari, M. (2009). Explicit model-predictive control of a pwm inverter with an LCL filter. *Industrial Electronics, IEEE Transactions on*, 56(2), 389–399.

Martello, S. and Toth, P. (1987). Linear assignment problems. In *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, 259–282. North-Holland.

Mattingley, J. and Boyd, S. (2009). *Convex Optimization in Signal Processing and Communications*, chapter Automatic Code Generation for Real-Time Convex Optimization. Cambridge University Press.

Mayne, D., Rawlings, J., Rao, C., and Scokaert, P. (2000). Constrained model predictive control: stability and optimality. *Automatica*, 26(6), 789–814.

Moehle, N. and Boyd, S. (2015). A perspective-based convex relaxation for switched-affine optimal control. *Systems & Control Letters*, 86, 34–40.

Mönnigmann, M. and Kastsian, M. (2011). Fast explicit model predictive control with multiway trees. In *Proceedings of the 18th IFAC World Congress*, 1356–1361.

Nedelcu, V., Necoara, I., and Quoc, D.Q. (2014). Computational complexity of inexact gradient augmented lagrangian methods: Application to constrained MPC. *SIAM Journal on Control and Optimization*, 52(5), 3109–3134.

Nesterov, Y. (1983). A method for solving a convex programming problem with rate of convergence $O(1/k^2)$. *Doklady Mathematics*, 269(3), 543–547.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition.

Nocedal, J., Wächter, A., and Waltz, R.A. (2009). Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4), 1674–1693.

Noga, R., de Prada, C., Ohtsuka, T., Blanco, E., and Casas, J. (2014). Non-linear moving horizon state estimation and control for the superfluid helium cryogenic circuit at the large Hadron Collider. In *53rd IEEE Conference on Decision and Control*, 3530–3535.

O'Donoghue, B. and Candes, E. (2015). Adaptive Restart for Accelerated Gradient Schemes. *Foundations of Computational Mathematics*, 15(3), 715–732.

Ohtsuka, T. and Kodama, A. (2002). Automatic code generation system for nonlinear receding horizon control. *Transactions of the Society of Instrument and Control Engineers*, 38(7), 617–623.

Ohtsuka, T. (2004). A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4), 563–574.

Ortner, P. and del Re, L. (2007). Predictive Control of a Diesel Engine Air Path. *IEEE Transactions on Control Systems Technology*, 15(3), 449–456.

Patrinos, P. and Bemporad, A. (2012). An accelerated dual gradient-projection algorithm for linear model predictive control. In *Proc. IEEE 51st Conf. on Decision and Control (CDC)*, 662–667.

Patrinos, P., Sopasakis, P., and Sarimveis, H. (2011). A global piecewise smooth newton method for fast large-scale model predictive control. *Automatica*, 47, 2016–2022.

Qin, S. and Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11, 733–764.

Quirynen, R., Gros, S., Houska, B., and Diehl, M. (2017). Lifted collocation integrators for direct optimal control in ACADO toolkit. *Mathematical Programming Computation*. (under review, preprint available at Optimization Online).

Rao, C., Wright, S., and Rawlings, J. (1998). Application of Interior-Point Methods to Model Predictive Control. *Journal of Optimization Theory and Applications*, 99, 723–757.

Rawlings, J. and Mayne, D. (2009). *Model Predictive Control: Theory and Design*. Nob Hill.

Richalet, J., Rault, A., Testud, J., and Papon, J. (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, 14, 413–428.

Richter, S., Morari, M., and Jones, C.N. (2011). Towards computational complexity certification for constrained

MPC based on lagrange relaxation and the fast gradient method. In *Conference on Decision and Control and European Control Conference (CDC-ECC)*, 5223–5229.

Rodriguez, J., Kazmierkowski, M.P., Espinoza, J.R., Zanchetta, P., Abu-Rub, H., Young, H.A., and Rojas, C.A. (2013). State of the art of finite control set model predictive control in power electronics. *IEEE Transactions on Industrial Informatics*, 9(2), 1003–1016.

Sager, S. (2005). *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg. ISBN 3-89959-416-9.

Sager, S., Bock, H.G., and Diehl, M. (2012). The integer approximation error in mixed-integer optimal control. *Mathematical Programming (Series A)*, 133, 1–23.

Sahinidis, N.V. (2014). *BARON: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual.

Sargent, R. and Sullivan, G. (1978). The development of an efficient optimal control package. In J. Stoer (ed.), *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*. Springer, Heidelberg.

Seguchi, H. and Ohtsuka, T. (2003a). Nonlinear Receding Horizon Control of an Underactuated Hovercraft. *International Journal of Robust and Nonlinear Control*, 13(3–4), 381–398.

Seguchi, H. and Ohtsuka, T. (2003b). Nonlinear receding horizon control of an underactuated hovercraft. *International journal of robust and nonlinear control*, 13(3-4), 381–398.

Stathopoulos, G., Shukla, H., Szűcs, A., Pu, Y., and Jones, C.N. (2016). Operator Splitting Methods in Control. *Foundations and Trends(R) in Systems and Control*, 3(3), 249–362.

Takapoui, R., Moehle, N., Boyd, S., and Bemporad, A. (2016). A simple effective heuristic for embedded mixed-integer quadratic programming. In *Proceedings of the American Control Conference, Boston, MA, USA*.

Tøndel, P., Johansen, T., and Bemporad, A. (2003). An Algorithm for Multi-Parametric Quadratic Programming and Explicit MPC Solutions. *Automatica*, 39, 489–497.

Torrisi, F. and Bemporad, A. (2004). HYSDEL – a tool for generating computational hybrid models for analysis and synthesis problems. *IEEE Transactions on Control Systems Technology*, 12(2), 235–249.

Tran-Dinh, Q., Savorgnan, C., and Diehl, M. (2012). Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization. *SIAM J. Optimization*, 22(4), 1258–1284.

Tseng, P. (2008). On accelerated proximal gradient methods for convex-concave optimization. URL `http://www.mit.edu/~dimitrib/PTseng/papers/apgm.pdf`.

Ullmann, F. (2011). *FiOrdOs: A MATLAB Toolbox for C-Code Generation for First Order Methods*. Master thesis, ETH Zurich, Switzerland.

van Duijkeren, N., Verschueren, R., Pipeleers, G., Diehl, M., and Swevers, J. (2016). Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation. In *Proceedings of the European Control Conference (ECC)*.

Vargas-Villamil, F.D. and Rivera, D.E. (2000). Multilayer optimization and scheduling using model predictive control: application to reentrant semiconductor manufacturing lines. *Computers & Chemical Engineering*, 24(8), 2009–2021.

Verschueren, R., Bruyne, S.D., Zanon, M., Frasch, J.V., and Diehl, M. (2014). Towards time-optimal race car driving using nonlinear MPC in real-time. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2505–2510.

Verschueren, R., van Duijkeren, N., Quirynen, R., and Diehl, M. (2016). Exploiting convexity in direct optimal control: a sequential convex quadratic programming method. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*.

Vukov, M., Loock, W.V., Houska, B., Ferreau, H., Swevers, J., and Diehl, M. (2012). Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In *Proceedings of the American Control Conference, Montreal, Canada*, 6264–6269.

Vukov, M., Gros, S., Horn, G., Frison, G., Geebelen, K., Jørgensen, J.B., Swevers, J., and Diehl, M. (2015). Real-time nonlinear MPC and MHE for a large-scale mechatronic application. *Control Engineering Practice*, 45, 64–78.

Wächter, A. and Biegler, L.T. (2006a). Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. *SIAM Journal on Optimization*, 16, 1–31.

Wächter, A. and Biegler, L.T. (2006b). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

Wang, Y. and Boyd, S. (2010). Approximate dynamic programming via iterated bellman inequalities.

Widd, A., Ekholm, K., Tunestal, P., and Johansson, R. (2009). Experimental evaluation of predictive combustion phasing control in an HCCI engine using fast thermal management and VVA. In *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*, 334–339. IEEE.

Wills, A., Bates, D., Fleming, A., Ninness, B., and Moheimani, S. (2005). Application of MPC to an active structure using sampling rates up to 25kHz. 3176–3181. 44th IEEE Conference on Decision and Control and European Control Conference, Seville, Spain.

Wolsey, L.A. and Nemhauser, G.L. (1988). *Integer and Combinatorial Optimization*. John Wiley & Sons.

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2016). FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control: Special Issue on MPC Algorithms and Applications*. (accepted for publication).

Zavala, V.M. and Biegler, L. (2009). The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45, 86–93.