Embedded Online Optimization for Model Predictive Control at Megahertz Rates

Juan L. Jerez, *Student Member, IEEE,* Paul J. Goulart, Stefan Richter,
George A. Constantinides, *Senior Member, IEEE,* Eric C. Kerrigan, *Member, IEEE,*
and Manfred Morari, *Fellow, IEEE*

***Abstract*—Faster, cheaper, and more power efficient optimization solvers than those currently possible using general-purpose techniques are required for extending the use of model predictive control (MPC) to resource-constrained embedded platforms. We propose several custom computational architectures for different first-order optimization methods that can handle linear-quadratic MPC problems with input, input-rate, and soft state constraints. We provide analysis ensuring the reliable operation of the resulting controller under reduced precision fixed-point arithmetic. Implementation of the proposed architectures in FPGAs shows that satisfactory control performance at a sample rate beyond 1 MHz is achievable even on low-end devices, opening up new possibilities for the application of MPC on embedded systems.**

***Index Terms*—Predictive control of linear systems, embedded systems, optimization algorithms**

I. Introduction

Model predictive control (MPC) provides a systematic approach for handling physical constraints for automatic control of cyber-physical systems [1], [2], often leading to improved control performance and reduced tuning effort for new applications. However, the intense computational demands imposed by MPC precludes its use in applications that could benefit considerably from its advantages, especially in those that have fast required response times and in those that must run on resource-constrained, embedded computing platforms with low cost or low power requirements.

For linearly constrained MPC problems of low dimensionality, one can partially avoid this computational burden by precomputing the solution map offline using multi-parametric programming [3]. In this case, the online controller implementation consists only of region search and table look-up procedures. Further work integrating the design of the solution map and embedded circuits has further increased the efficiency in performing these operations [4]. However, this approach quickly becomes impractical for larger problems, mainly due to substantial memory requirements, forcing a return to online optimization methods.

Recently, there has been significant interest in using first-order methods, both in the primal [5] and dual domains [6]–[9], for the online solution of linear-quadratic MPC problems. Compared to other solution methods for quadratic programs (QPs) (e.g. active-set or interior-point schemes), first-order methods do not require the solution of a linear system of equations at every iteration, which is often a limiting factor for embedded platforms with modest computational capability. This feature, coupled with the observation that medium-accuracy solutions are often sufficient for good control performance [10], make first-order methods promising candidates for efficient, low cost MPC. In addition, first-order methods have certain features that make them amenable to fixed-point implementation, they can be efficiently parallelized, and their simplicity invites analysis that can guide low-level implementation choices for further efficiency gains.

There have been several recent efforts to translate innovation in optimization algorithms into practical solvers customized for MPC problems. In terms of software, [11], [12] and [13] describe automatic state-of-the-art code generators for interior-point and first-order solvers, respectively, whereas [14] describes a widely used active-set based solver. In all cases, embedded applications were the primary target, although the solvers are implemented using double precision floating-point arithmetic which is generally not available or is very expensive in embedded computing platforms. In terms of hardware, [15]–[17] describe different custom computing architectures for both interior-point and active-set methods using reduced floating-point arithmetic in field programmable gate arrays (FPGAs), reporting minor speed-ups or use of expensive devices to provide significant acceleration. Although there has been some progress in accelerating the core component of these algorithms – solvers for linear equations – using fixed-point arithmetic [18], extending these results to the other aspects of interior-point or active-set algorithms remains challenging.

Summary of contribution

In this paper we focus on practical and theoretical issues for efficient implementation of optimization-based control systems on low cost embedded platforms.

1) *Architectures*: We present a set of parameterized automatic generators of custom computing architectures for solving different types of MPC problems. For input-constrained problems, we describe architectures for Nesterov's fast gradient method (first described in the preliminary publication [19]). For state-constrained problems we describe architectures based on the alternating direction method of multipliers (ADMM). Even if these

Juan L. Jerez , Paul J. Goulart, Stefan Richter and Manfred Morari are with the Automatic Control Laboratory, ETH Zürich, 8092 Zürich, Switzerland, juanj|pgoulart|richters|morari@control.ee.ethz.ch

George A. Constantinides is with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ, United Kingdom, gac1@imperial.ac.uk

Eric C. Kerrigan is with the Department of Electrical and Electronic Engineering and the Department of Aeronautics, Imperial College London, SW7 2AZ, United Kingdom, e.kerrigan@imperial.ac.uk

methods are conceptually very different, they share the same computational patterns and similar computing architectures can be used to implement them efficiently. These architectures are extended to support warm starting procedures and the projection operations required in the presence of soft constraints.

2) *Analysis*: Since for a reliable operation using fixed-point arithmetic it is crucial to prevent overflow errors, we derive theoretical results that guarantee the absence of overflow in all variables of the fast gradient method. Furthermore, we present an error analysis of both the fast gradient method and ADMM under (inexact) fixed-point computations in a unified framework. This analysis underpins the numerical stability of the methods for hardware implementations and can be used to determine *a priori* the minimum number of bits required to achieve a given solution accuracy specification, resulting in minimal resource usage.
3) *Implementation*: We derive a set of design rules for efficient implementation of the proposed methods, such as a scaling procedure for accelerating the convergence of ADMM and criteria for determining the size of the Lagrange multipliers. The proposed architectures are characterized in terms of the achievable performance as a function of the amount of resources available. As a proof of concept, generated solver instances are demonstrated for several linear-quadratic MPC problems, reporting achievable controller sampling rates in excess of 1 MHz, while the controller can be implemented on a low cost embeddable device.

Outline

The paper is organized as follows: After a brief summary of the general MPC formulation and the different first-order methods in Sections II and III, we focus on the fixed-point analysis in Section IV. We follow with the hardware architectures and performance evaluation in Sections V and VI.

II. Soft-Constrained Model Predictive Control Setup

Throughout, we address control of a discrete-time linear time-invariant (LTI) system in the form

$$x^+ = Ax + Bu, \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ is the system state, $u \in \mathbb{R}^{n_u}$ is the system input and x^+ denotes the state at the next sampling instant. The overall design goal is to construct a time-invariant (possibly nonlinear) static state feedback controller $\mu : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ such that $u = \mu(x)$ stabilizes the system (1) while simultaneously satisfying a collection of state and input constraints in the time domain.

In standard design methods for constructing linear controllers for systems in the form (1), the bulk of the computational effort is spent *offline* in identifying a suitable controller, whose *online* implementation has minimal computing requirements. The inclusion of state and input constraints renders most such design methods unsuitable.

A now standard alternative is to use MPC [1], [2], which moves the bulk of the required computational effort *online* and which addresses directly the system constraints. At every sampling instant, given an estimate or measurement of the current state of the plant x, an MPC controller solves a constrained N-stage optimal control problem in the form

$$\begin{aligned} J^*(x) = \min\ & \frac{1}{2}x_N^T Q_N x_N + \frac{1}{2}\sum_{k=0}^{N-1} x_k^T Q x_k + u_k^T R u_k \\ & + 2x_k^T S u_k + \sum_{k=1}^{N}\left(\sigma_1 \cdot \mathbf{1}^T \delta_k + \sigma_2 \cdot \|\delta_k\|_2^2\right) \\ \text{subject to}\ \ & x_0 = x, \\ & x_{k+1} = A_d x_k + B_d u_k, \quad k = 0, 1, \ldots, N-1, \\ & u_k \in \mathbb{U}, \qquad\qquad k = 0, 1, \ldots, N-1, \\ & (x_k, \delta_k) \in \mathbb{X}_\Delta, \qquad\quad k = 0, 1, \ldots, N. \end{aligned} \tag{2}$$

If an optimal input sequence $\{u_i^*(x)\}_{i=0}^{N-1}$ and state trajectory $\{x_i^*(x)\}_{i=0}^{N}$ exists for this problem given the initial state x, then an MPC controller can be implemented by applying the control input $u = u_0^*(x)$.

We will assume throughout that the system input constraint set $\mathbb{U}$ is defined as a set of interval constraints $\mathbb{U} := \{u \mid u_{\min} \le u \le u_{\max}\}$. We assume that the system states have both free (denoted by $x_{\mathcal{F}}$ with index set $\mathcal{F}$), hard-constrained (index set $\mathcal{B}$) and soft-constrained (index set $\mathcal{S}$) components, i.e. the set $\mathbb{X}_\Delta$ in (2) is defined as

$$\mathbb{X}_\Delta = \left\{(x, \delta) \in \mathbb{R}^{n_x} \times \mathbb{R}_+^{|\mathcal{S}|} \;\middle|\; \begin{array}{l} x_{\mathcal{F}} \text{ free}, \ x_{\min} \le x_{\mathcal{B}} \le x_{\max}, \\ |x_i - x_{\mathrm{c},i}| \le r_i + \delta_i, \ i \in \mathcal{S} \end{array}\right\}, \tag{3}$$

with $x_{\mathrm{c},i} \in \mathbb{R}$ being the center of the interval constraint of radius $r_i > 0$ for a soft-constrained state component. The index sets $\mathcal{F}, \mathcal{B}$ and $\mathcal{S}$ are assumed to be pairwise disjoint and to satisfy $\mathcal{F} \cup \mathcal{B} \cup \mathcal{S} = \{1, 2, \ldots, n_x\}$.

We assume throughout that the penalty matrices $Q \in \mathbb{R}^{n_x \times n_x}$, $Q_N \in \mathbb{R}^{n_x \times n_x}$ are positive semidefinite, $R \in \mathbb{R}^{n_u \times n_u}$ is positive definite, and $S \in \mathbb{R}^{n_x \times n_u}$ is chosen such that the objective function in (2) is jointly convex in the states and inputs. There is by now a considerable body of literature [2], [20] describing conditions on the penalty matrices and/or horizon length N sufficient to ensure that the resulting MPC controller is stabilizing (even when no terminal state constraints are imposed[1]), and we do not address this point further. For stability conditions for soft-constrained problems, the reader is referred to [21] and [22] and the references therein. In the presence of numerical error, the sub-optimality can be interpreted as an additive bounded state disturbance. See [23] for a detailed investigation of stability properties under this scenario.

If the soft-constrained index set $\mathcal{S}$ is nonempty, then a linear-quadratic penalty on the auxiliary variables $\delta_k \in \mathbb{R}_+^{|S|}$, weighted by positive scalars (σ_1, σ_2), can be added to the objective. In practice, soft constraints are a common measure

[1] In the presence of polyhedral or ellipsoidal terminal constraints, the state-constrained methods described in this paper can still be used by adding a small number of ancillary variables.

to avoid infeasibility of the MPC problem (2) in the presence of disturbances. However, there also exist hard state constraints that can always be enforced and cannot lead to infeasibility, such as state constraints arising from remodeling of input-rate constraints. For the sake of generality we address both types of state constraints in this paper.

If σ_1 is chosen large enough, then the optimization problem (2) corresponds to an *exact penalty* reformulation of the associated hard-constrained problem (i.e. one in which the optimal solution of (2) maintains $\delta_k = 0$ if it is possible to do so). An exact penalty formulation preserves the optimal behavior of the MPC controller when all constraints can be enforced. We first characterize conditions under which a soft constraint penalty function for a convex optimization problem is exact.

Theorem 1 ([24, Prop. 5.4.5])**.** *Consider the convex problem*

$$f^* := \min_{z\in\mathbb{Q}} f(z) \quad \text{subject to } g_j(z) \le 0\,, \quad j = 1,2,\dots,r, \tag{4}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1,\dots,r$, are convex, real-valued functions and $\mathbb{Q}$ is a closed convex subset of $\mathbb{R}^n$. Assume that an optimal solution z^* exists with $f(z^*) = f^*$, strong duality holds and an optimal Lagrange multiplier vector $\mu^* \in \mathbb{R}^r_+$ for the inequality constraints exists.

i) If $\sigma_1 \ge \|\mu^*\|_\infty$ and $\sigma_2 \ge 0$, then

$$f^* = \min_{z\in\mathbb{Q}} f(z) + \sum_{j=1}^{r} \left(\sigma_1 \cdot \delta_j + \sigma_2 \cdot \delta_j^2\right) \quad \text{subject to } g_j(z) \le \delta_j, \quad \delta_j \ge 0, \quad j = 1,2,\dots,r. \tag{5}$$

ii) If $\sigma_1 > \|\mu^*\|_\infty$ and $\sigma_2 \ge 0$, the set of minimizers of the penalty reformulation in (5) *coincides with the set of minimizers of the original problem in* (4)*.*

Remark 1. *In the context of the MPC problem* (2)*, the penalty reformulation is exact if the penalty parameter σ_1 is chosen to be greater than the largest Lagrange multiplier for any constraint $|x_i - x_{c,i}| \le r_i$, $i \in \mathcal{S}$, over all feasible initial states x. In general, this bound is unknown* a priori *and is treated as a tuning parameter in the control design. The quadratic penalty parameter σ_2 need not be nonzero for such a penalty formulation to be exact, but the inclusion of a nonzero quadratic term is necessary for our numerical stability results under fixed-point arithmetic in Section IV.*

For the sake of notational simplicity, the results of this paper are presented with reference to the optimal control problem in regulator form in (2). However, all of our results generalize easily to setpoint tracking problems.

III. First-Order Solution Methods

We next describe two different first-order optimization methods for solving the optimal control problem (2) efficiently. In particular, we apply the primal fast gradient method (FGM) in cases where only input-constraints are present, and a dual method based on the alternating direction method of multipliers (ADMM) for cases in which both state- and input-constraints are present.

A. Input-Constrained MPC Using the Fast Gradient Method

The fast gradient method is an iterative solution method for smooth convex optimization problems first published by Nesterov in the early 80s [25]. The method can be applied to the solution of MPC problem (2) if the future state variables x_i are eliminated by expressing them as a function of the initial state, x, and the future input sequence (so-called *condensing* [1]), resulting in the problem

$$f^*(x) = \min_z f(z;x) := \frac{1}{2} z^T H_F z + z^T \Phi x \quad \text{subject to } z \in \mathbb{K}, \tag{6}$$

where $z := (u_0,\dots,u_{N-1}) \in \mathbb{R}^n$, $n = Nn_u$, the Hessian $H_F \in \mathbb{R}^{n\times n}$ is positive definite under the assumptions in Section II, and the feasible set is given as $\mathbb{K} := \mathbb{U} \times \dots \times \mathbb{U}$. The current state only enters the gradient of the linear term of the objective through the matrix $\Phi \in \mathbb{R}^{n\times n_x}$. See [1] for details on the construction of the matrices.

We consider the *constant step scheme II* of the fast gradient method in [26, §2.2.3]. Its algorithmic scheme for the solution of (6), optimized for parallel execution on parallel hardware, is given in Algorithm 1. Note that the state-independent terms $(I - \frac{1}{L}H_F)$, $\frac{1}{L}\Phi$ and $(1+\beta)$ can all be computed offline and that the product $\frac{1}{L}\Phi x$ must only be evaluated once. The core operations in Algorithm 1 are the evaluation of the gradient (implicit in line 2) and the projection operator of the feasible set, $\pi_{\mathbb{K}}$, in line 3. Since for our application the set $\mathbb{K}$ is the direct product of the N n_u-dimensional sets $\mathbb{U}$, it suffices to consider N independent projections that can be performed in parallel. For the specific case of a box constraint on the control input, every such projection corresponds to n_u scalar projections on intervals, each computable analytically. In this case, the fast gradient method requires only multiplication and addition, which are considerably faster and use significantly less resources than division when implemented using digital circuits.

It can be inferred from [26, Theorem 2.2.3] that for every state x, Algorithm 1 generates a sequence of iterates $\{z_i\}_{i=1}^{I_{\max}}$ such that the residuals $f(z_i;x) - f^*(x)$ are bounded by

$$\min\left\{\left(1 - \sqrt{\frac{1}{\kappa}}\right)^i, \frac{4\kappa}{(2\sqrt{\kappa}+i)^2}\right\} \cdot 2\big(f(z_0;x) - f^*(x)\big), \tag{7}$$

for all $i = 0,\dots,I_{\max}$, where κ denotes the condition number of f, or an upper bound of it, given by $\kappa = L/\mu$, where L and μ are a Lipschitz constant for the gradient of f and convexity parameter of f, respectively. Note that the convexity parameter for a strongly convex quadratic objective function as in (6) corresponds to the minimum eigenvalue of H_F. Based on this convergence result, which states that the bound exhibits the best of a linear and a sublinear rate, one can derive a certifiable and practically relevant iteration bound $I_{\max}$ such that the final residual is guaranteed to be within a specified level of suboptimality for all initial states arising from a bounded set [5]. It can further be proved that there is no other variant

Algorithm 1 Fast gradient method for the solution of MPC problem (6) at state x (*optimized for parallel hardware*)

Require: Initial iterate $z_0 \in \mathbb{K}$, $y_0 = z_0$, upper (lower) bound L ($\mu > 0$) on maximum (minimum) eigenvalue of Hessian H_F, step size $\beta = \left(\sqrt{L} - \sqrt{\mu}\right)/\left(\sqrt{L} + \sqrt{\mu}\right)$

1: **for** $i = 0$ to $I_{\max} - 1$ **do**
2: $\quad t_i := (I - \frac{1}{L}H_F)y_i - \frac{1}{L}\Phi x$
3: $\quad z_{i+1} := \pi_{\mathbb{K}}(t_i)$
4: $\quad y_{i+1} := (1+\beta)z_{i+1} - \beta z_i$
5: **end for**

of a gradient method with better theoretical convergence [26, Thm. 2.2.2], i.e. the fast gradient method is an *optimal* gradient method.

The fast gradient method is particularly attractive for application to MPC in embedded control system design due both to the relative ease of implementation and to the availability of strong performance certification guarantees. However, its use is limited to cases in which the projection operation $\pi_{\mathbb{K}}$ is simple, e.g. in the case of box-constrained inputs. Unfortunately, the inclusion of state constraints changes the geometry of the feasible set $\mathbb{K}$ such that the projection subproblem is as difficult as the original problem, since the constraints are no longer separable in u_k. In the next section we therefore describe an alternative solution method in the dual domain that avoids these complications, though at the expense of some of the strong certification advantages.

B. Input- and State-Constrained MPC Using ADMM

In the presence of state constraints, if one requires both Q and Q_N to be positive definite, the fast gradient method can be used again to solve the dual problem via Lagrange relaxation of the equality constraints [6]. However, in this case the dual function is *not* strongly concave and consequently the convergence rate is severely affected (from linear to sublinear). A quadratic regularizing term can be added to the Lagrangian to improve convergence (the so-called *method of multipliers*), but this prevents the use of distributed operations for computing the gradient of the dual function, adding a significant computational overhead. We therefore seek an alternative approach in the dual domain.

For dual problems we do *not* work in the condensed format (6), but rather maintain the state variables x_k in the vector of decision variables $z := (u_0, \ldots, u_{N-1}, x_0, \delta_0, \ldots, x_N, \delta_N) \in \mathbb{R}^n$, $n = N(n_u + n_x + |\mathcal{S}|) + n_x + |\mathcal{S}|$, resulting in the problem

$$\begin{aligned} f^*(x) = \min_z f(z;x) := \frac{1}{2} z^T H_A z + z^T h \\ \text{subject to } z \in \mathbb{K},\ Fz = b(x). \end{aligned} \tag{8}$$

The affine constraint $Fz = b(x)$ models the dynamic coupling of the states x_k and u_k via the state update equation (1), and is at the root of the difficulty in projecting the variables z onto the constraints in the fast gradient method.

The alternating direction method of multipliers (ADMM) [27] partitions the optimization variables into two (or more) groups to maintain the possibility of decoupled projection. In applying ADMM to the specific problem (6), we maintain an additional copy y of the original decision variables z and solve the problem

$$\begin{aligned} f^*(x) = \min_{z,y} f(z,y;x) := \frac{1}{2} y^T H_A y + y^T h \\ + I_{\mathbb{A}}(y;x) + I_{\mathbb{K}}(z) + \frac{\rho}{2}\|y - z\|^2 \end{aligned} \tag{9}$$

$$\text{subject to } z = y, \tag{10}$$

where $(z, y) \in \mathbb{R}^{2n}$ contain copies of all input, state and slack variables. The functions $I_{\mathbb{A}} : \mathbb{R}^n \times \mathbb{R}^{n_x} \to \{0, +\infty\}$ and $I_{\mathbb{K}} : \mathbb{R}^n \to \{0, +\infty\}$ are indicator functions for the sets described by the equality and inequality constraints, respectively, e.g.

$$I_{\mathbb{A}}(y;x) := \begin{cases} 0 & \text{if } Fy = b(x)\,, \\ +\infty & \text{otherwise}\,, \end{cases} \tag{11}$$

where $\mathbb{K} := \mathbb{U} \times \ldots \times \mathbb{U} \times \mathbb{X}_\Delta \times \ldots \times \mathbb{X}_\Delta$. The current state x enters the optimization problem through (11). The inclusion of the regularizing term $(\rho/2)\|y - z\|^2$ has no impact on the solution to (9) (equivalently (8)) due to the compatibility constraint $y = z$, but it does allow one to drop the smoothness and strong convexity conditions on the objective function, so that one can solve control problems with more general cost functions such as those with 1- or ∞-norm stage costs.

Note that there are many possible techniques for copying and partitioning of variables in ADMM. In the context of optimal control, the choice given in (9) results in attractive computational structures [28].

The dual problem for (9) is given by

$$\begin{aligned} \max_{\nu} \inf_{z,y} L_\rho(z,y,\nu) := \frac{1}{2} y^T H_A y + y^T h + I_{\mathbb{A}}(y;x) \\ + I_{\mathbb{K}}(z) + \nu^T (y - z) + \frac{\rho}{2}\|y - z\|^2\,. \end{aligned}$$

ADMM solves this dual problem by repeatedly carrying out the steps

$$y_{i+1} := \arg\min_y L_\rho(z_i, y, \nu_i)\,, \tag{12a}$$

$$z_{i+1} := \arg\min_z L_\rho(z, y_{i+1}, \nu_i)\,, \tag{12b}$$

$$\nu_{i+1} := \nu_i + \rho(y_{i+1} - z_{i+1})\,. \tag{12c}$$

The parameter ρ can be any positive number to ensure convergence. There are at present no universally accepted rules for selecting the value of the penalty parameter however, and it is typically treated as a tuning parameter during implementation. See [29], [30] for a more detailed discussion.

Our overall algorithmic scheme for ADMM for the solution of (9) based on the sequence of operations (12a)–(12c), optimized for parallel execution on parallel hardware, is given in Algorithm 2. The core computational tasks are the equality-constrained optimization problem (12a) and the inequality-constrained, but separable, optimization problem (12b).

In the case of the equality-constrained minimization step (12a), a solution can be computed from the KKT con-

Algorithm 2 ADMM for the solution of MPC problem (6) at state x (*optimized for parallel hardware*)

Require: Initial iterate $z_0 = z^{*-}$, $\nu_0 = \nu^{*-}$, where z^{*-} and ν^{*-} are the shifted solutions at the previous time instant (see Section V), and ρ is a constant power of 2.

1: **for** $i = 0$ to $I_{\max} - 1$ **do**
2: $\quad y_{i+1} := M_{11}(-h + \rho z_i - \nu_i) + M_{12} b(x)$
3:
4: $\quad \nu_{i+1} := \rho y_{i+1} + \nu_i - \rho z_{i+1}$
5: **end for**

ditions by solving the linear system

$$\begin{bmatrix} H_A + \rho I & F^T \\ F & 0 \end{bmatrix} \begin{bmatrix} y_{i+1} \\ \lambda_{i+1} \end{bmatrix} = \begin{bmatrix} -h - \nu_i + \rho z_i \\ b(x) \end{bmatrix}.$$

Note that only the vector y_{i+1}, and not the multiplier λ_{i+1}, arising from the solution of this linear system is required for our ADMM method. The most efficient method to solve for y_{i+1} is to invert the (fixed) KKT matrix *offline*, i.e. to compute

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{12}^T & M_{22} \end{bmatrix} = \begin{bmatrix} H_A + \rho I & F^T \\ F & 0 \end{bmatrix}^{-1},$$

and then to obtain y_{i+1} *online* from $y_{i+1} = M_{11}(-h - \nu_i + \rho z_i) + M_{12} b(x)$ as in Line 2 of Algorithm 2. Observe that the product $M_{12} b(x)$ needs to be evaluated only once, and that this matrix is always invertible when $\rho > 0$ since F has full row rank.

The inequality-constrained minimization step (12b) results in the projection operation in Line 3 of Algorithm 2. In the presence of soft state constraints, this operation requires independent projections onto a truncated two-dimensional cone, which can be efficiently parallelized and require no divisions. We describe efficient implementations of this projection operation in parallel hardware in Section V.

This variant of ADMM is known to converge; see [31, §3.4; Prop. 4.2] for general convergence results. More recently, sublinear and linear convergence rates were established. See [32], [33] for more details.

C. ADMM, Lagrange multipliers and soft constraints

Despite its generally excellent empirical performance, ADMM can be observed to converge very slowly in certain cases. In particular, for MPC problems in the form (6), convergence may be very slow in those cases where there is a large mismatch between the magnitude of the optimal Lagrange multipliers ν^* for the equality constraint (10) and the magnitude of the primal iterates (z_i, y_i). The reason is evident from the ADMM multiplier update step (12c); the existence of very large optimal multipliers ν^* necessitates a large number of ADMM iterations when the difference $(z_i - y_i)$ remains small at each iteration and $\rho \approx 1$.

This effect is of particular concern for MPC problem instances with soft constraints. If one denotes by z_δ those components of z associated with the slack variables $\{\delta_1, \ldots, \delta_N\}$ (with similar notation for y_δ), then the objective function (9) features a term $\sigma_1 \cdot \mathbf{1}^T y_\delta$, with the exact penalty term σ_1 typically very large. The equality constraints (10) include the matching condition $z_\delta - y_\delta = 0$, with associated Lagrange multiplier ν_δ. Recalling the usual sensitivity interpretation of the optimal multiplier ν_δ^*, one can conclude that $\nu_\delta^* \approx \sigma_1 \cdot \mathbf{1}$ in the absence of unusual problem scaling[2].

For soft constrained problems, we avoid this difficulty by rescaling those components of the matching condition (10) to the equivalent condition $(1/\sigma_1)(z_\delta - y_\delta) = 0$, which results in a rescaling of the associated optimal multipliers to $\nu_\delta^* \approx 1$. The aforementioned convergence difficulties due to excessively large optimal multipliers are then avoided.

IV. Fixed-Point Aspects of First-Order Solution Methods

In this section we first motivate the use of fixed-point arithmetic from a hardware efficiency perspective and then isolate potential error sources under this arithmetic. We concentrate on two types of errors. For *overflow errors* we provide analysis to guarantee that they cannot occur in the fast gradient method, whereas for *arithmetic round-off errors* we prove that there is a converging upper bound on the total incurred error in either of the two methods. The results we obtain hold under the assumptions in Section IV-B and guarantee reliable operation of first-order methods on fixed-point platforms.

A. Fixed-Point Arithmetic and Error Sources

Modern computing platforms must allow for a wide range of applications that operate on data with potentially large dynamic range, i.e. the ratio of the smallest to largest number to be represented. For general purpose computing, *floating-point* arithmetic provides the necessary flexibility. A floating-point number consists of a sign bit, a mantissa, and an exponent value that moves the binary point with respect to the mantissa. The dynamic range grows doubly exponentially with the number of exponent bits, making it possible to represent a wide range of numbers with a relatively small number of bits. However, because two operands can have different exponents, it is necessary to perform *denormalization* and *normalization* operations before and after every addition or subtraction, leading to increased resource usage and long arithmetic delays.

In contrast, hardware platforms employing *fixed-point* numbers use a fixed number of bits for the integer and fraction fields, i.e. the exponent does not vary and does not need to be stored. Fixed-point computations are the same as with integer arithmetic, hence the digital circuitry is simple and fast, leading to greater power efficiency and significant potential for acceleration via extra parallelization in a custom hardware implementation. For instance, in a typical modern FPGA platform [34] fixed-point addition takes one clock cycle, whereas a single precision floating-point adder would require 14 cycles while using one order of magnitude more resources for the same number of bits.

[2]If one sets the regularization parameter $\rho = 0$ in (9) and $\sigma_2 = 0$, then it can be shown that this approximation becomes exact.

The benefits of fixed-point arithmetic motivate its use in first-order methods to realize fast and efficient implementations of Algorithms 1 and 2 on FPGAs or other low cost and low power devices with no floating-point support, such as embedded microcontrollers, fixed-point digital signal processors (DSPs) or programmable logic controllers (PLCs). However, reduced precision representations and fixed-point computations incur several types of errors that must be accounted for. These include:

Quantization Errors: Finite representation errors arise when converting the problem and algorithm data from high precision to reduced precision data formats. Potential consequences include loss of problem convexity, change of optimal solution and a lack of feasibility with respect to the original problem.

Overflow Errors: Overflow errors occur whenever the number of bits for the integer part in the fixed-point representation is too small, and can cause unpredictable behavior of the algorithm.

Arithmetic Errors: Unlike with floating-point arithmetic, fixed-point addition and subtraction operations involve no round-off error provided there is no overflow and the result has the same number of fraction bits as the operands [35]. For multiplication, the exact product of two numbers with b fraction bits can be represented using $2b$ fraction bits, hence a b-bit truncation of a 2's complement number incurs a round-off error bounded from below by -2^{-b}. Recall that in 2's complement arithmetic, truncation incurs a negative error both for positive and negative numbers.

B. Notation and Assumptions

We will use $\hat{(\cdot)}$ throughout in order to distinguish quantities in a fixed-point representation from those in an exact representation and under exact arithmetic. Throughout, we assume for simplicity that all variables and problem data are represented using the same number of fraction bits b. We further assume that the feasible sets under finite precision satisfy $\widehat{\mathbb{K}} \subseteq \mathbb{K}$, so that solutions in fixed point arithmetic do not produce infeasibility in the original problem due to quantization error.

We conduct separate analyses of both overflow and arithmetic errors for the fast gradient method (Algorithm 1) and ADMM (Algorithm 2). In both cases, the central requirement is to choose the number of fraction bits b large enough to ensure satisfactory numerical behavior. We therefore employ two different sets of assumptions depending on the numerical method in question:

Assumption 1 (Fast Gradient Method / Algorithm 1)**.** *The number of fraction bits b and a constant $c \geq 1$ are chosen large enough such that*

i) The matrix

$$H_n = \frac{1}{c \cdot \lambda_{\max}(\hat{H}_F)} \cdot \hat{H}_F,$$

has a fixed-point representation $\hat{H}_n$ with all of its eigenvalues in the interval $(0, 1]$, where $\hat{H}_F$ is the fixed-point representation of the Hessian H_F, with $\lambda_{\max}(\hat{H}_F)$ its maximum eigenvalue.

ii) The fixed-point step size $\hat{\beta}$ satisfies

$$1 > \hat{\beta} \geq \left(\sqrt{\kappa\big(\hat{H}_n\big)} - 1\right)\left(\sqrt{\kappa\big(\hat{H}_n\big)} + 1\right)^{-1} \geq 0\,,$$

where $\kappa(\hat{H}_n)$ is the condition number of $\hat{H}_n$.

Assumption 2 (ADMM / Algorithm 2)**.** *The number of fraction bits b is chosen large enough such that*

i) The matrix

$$\left(\begin{bmatrix}\hat{M}_{11} & \hat{M}_{12}\\ \hat{M}_{12}^T & \hat{M}_{22}\end{bmatrix}^{-1} - \begin{bmatrix}\rho I & \hat{F}^T\\ \hat{F} & 0\end{bmatrix}\right)$$

is positive semidefinite, where ρ is chosen such that it is exactly representable in b bits.

Observe that it is always possible to select b sufficiently large to satisfy all of the preceding assumptions, implying that the above conditions represent a lower bound on the number of fraction bits required in a fixed-point implementation of our two algorithms to ensure that our stability results are valid. Assumptions 1.(i) and 2.(i) ensure that the objective functions (6) (for the fast gradient method) and (9) (for ADMM) remain strongly convex and convex, respectively, despite any quantization error.

In the case of the fast gradient method, Assumption 1.(ii) guarantees that the *true* condition number of $\hat{H}_n$ is not underestimated, in which case the convergence result of the fast gradient method in (7) would be invalid. In fact, the assumption ensures that the effective condition number for the convergence result is given by

$$\kappa_n = \left(\frac{1+\hat{\beta}}{1-\hat{\beta}}\right)^2 \geq \kappa\big(\hat{H}_n\big). \tag{13}$$

C. Overflow Errors

In order to avoid overflow errors in a fixed-point implementation, the largest absolute values of the iterates' and intermediate variables' components must be known or upper-bounded *a priori* in order to determine the number of bits required for their *integer parts*. For the *static* problem data $(I - \hat{H}_n)$, $\hat{\Phi}_n$, $1 + \hat{\beta}$, $\hat{\beta}$, $\hat{M}_{11}$, or $\hat{M}_{12}$, the number of integer bits is easily determined by the maximum absolute value in each expression.

1) Overflow Error Bounds in the Fast Gradient Method:

In the case of the fast gradient method, it is possible to bound analytically the largest absolute values of all of the dynamic data, i.e. the variables that change with every iteration. We will denote by $\hat{\Phi}_n$ the fixed-point representation of

$$\Phi_n = \frac{1}{c \cdot \lambda_{\max}(\hat{H}_F)} \cdot \Phi.$$

We summarize the upper bounds on variables appearing in the fast gradient method in the following proposition:

Proposition 1. *If problem* (6) *is solved by the fast gradient method using the appropriately adapted Algorithm 1, then*

the largest absolute values of the iterates and intermediate variables are given by

$$\begin{aligned}\|\hat{z}_{i+1}\|_\infty &\le \bar{z} := \max\left\{\|\hat{z}_{\min}\|_\infty, \|\hat{z}_{\max}\|_\infty\right\},\\ \|\hat{y}_{i+1}\|_\infty &\le \bar{y} := \bar{z} + \hat{\beta}\|\hat{z}_{\max} - \hat{z}_{\min}\|_\infty,\\ \|(I-\hat{H}_n)\,\hat{y}_i\|_\infty &\le \bar{y}_{\textit{inter}} := \|I-\hat{H}_n\|_\infty \cdot \bar{y},\\ \|\hat{x}\|_\infty &\le \bar{x} := \max_{x\in\widehat{\mathbb{X}}_0}\|x\|_\infty,\\ \|\hat{\Phi}_n\hat{x}\|_\infty &\le \bar{h} := \|\hat{\Phi}_n\|_\infty \cdot \bar{x},\ \textit{and}\\ \|t_i\|_\infty &\le \bar{t} := \bar{y}_{\textit{inter}} + \bar{h},\end{aligned} \tag{14}$$

for all $i = 0, 1, \ldots, I_{\max}-1$. The set $\widehat{\mathbb{X}}_0$ is chosen such that for every state in exact arithmetic $x \in \mathbb{X}_0$ we have $\hat{x} \in \widehat{\mathbb{X}}_0$.

Proof. Follows from interval arithmetic and properties of the vector/matrix $\|\cdot\|_\infty$-norm. □

Note that the bound in (14) also applies for the intermediate elements/cumulative sums in the evaluation of the matrix-vector product. Observe that most of the bounds stated in Proposition 1 are tight.

2) Overflow Error Bounds in ADMM:

If problem (9) is solved using ADMM via Algorithm 2, then we are not aware of any general method to upper bound the Lagrange multiplier iterates ν_i analytically, and consequently are unable to establish analytic upper bounds on all expressions involving *dynamic* data. In this case, one must instead estimate the undetermined upper bounds through simulation and add a safety factor when allocating the number of integer bits. As a result, with ADMM, we trade analytical guarantees on numerical behavior for the capability to solve more general problems.

D. Arithmetic Round-Off Errors

We next derive an upper bound on the deviation of an optimal solution $\hat{z}^*$ produced via a fixed-point implementation of either Algorithm 1 or 2 from the optimal solutions produced from the same algorithms implemented using exact arithmetic. In both cases, we denote by $\hat{z}_i$ a *fixed-point* iterate. We wish to relate these iterates to the iterates z_i generated under *exact arithmetic*, by establishing a bound in the form

$$\|\hat{z}_i - z_i\| = \|\eta_i\| \le \Delta_i$$

with $\lim_{i\to\infty}\Delta_i$ finite, where $\eta_i := \hat{z}_i - z_i$ is the solution error attributable to arithmetic round-off error up to the i^{th} iteration. Consequently, we can show that inaccuracy in the computed optimal solution induced by arithmetic errors in either algorithm are bounded, which is a crucial prerequisite for reliable operation of first-order methods on fixed-point platforms.

In both cases, we use a control-theoretic approach based on standard Lyapunov methods to derive bounds on the solution error arising specifically from fixed-point arithmetic error. For simplicity of exposition, we consider only those errors arising from arithmetic errors and neglect quantization errors in the analysis. This choice does not alter substantively the results presented for either algorithm. Our approach is in contrast to (and more direct than) other approaches to error accumulation analysis in the fast gradient method such as [36], [37], which consider inexact gradient computations but do not address arithmetic round-off errors explicitly. In the case of ADMM, we are not aware of any existing results relating to error accumulation in fixed-point arithmetic.

1) Stability of Arithmetic Errors in the Fast Gradient Method:

We consider first the numerical stability of the fast gradient method, by examining in detail the arithmetic error introduced at each step of a fixed-point implementation of Algorithm 1.

At iteration i, the error in line 2 of Algorithm 1 is given by

$$\hat{t}_i - t_i = (I-\hat{H}_n)(\hat{y}_i - y_i) + \epsilon_{t,i}\,,$$

where $\epsilon_{t,i}$ is a vector of errors from the matrix-vector multiplication. Since there are n round-off errors in the computation of every component, $\epsilon_{t,i}$ is componentwise in the interval $[-n2^{-b}, 0]$.

For the projection in line 3, and recalling that $\widehat{\mathbb{K}} \subseteq \mathbb{K}$ is a box, no arithmetic error is introduced. Indeed, one can easily verify that the error $\hat{t}_i - t_i$ can only be reduced by projecting onto a box, i.e. if $\hat{t}_i$ and t_i are inside the feasible region the error remains, but if both quantities are saturated the corresponding component of the error goes to zero. This effect is modelled by multiplication with a diagonal matrix $\mathrm{diag}(\epsilon_{\pi,i})$, with $\epsilon_{\pi,i}$ componentwise in the interval $[0, 1]$.

Finally, in line 4, the error induced by fixed-point arithmetic is

$$\hat{y}_{i+1} - y_{i+1} = (1+\hat{\beta})\eta_{i+1} - \hat{\beta}\eta_i + \epsilon_{y,i}\,,$$

where two scalar-vector multiplications incur error $\epsilon_{y,i}$ with components in $[-2^{-b}, 2^{-b}]$ (addition *and* subtraction). Defining the initial error residual terms $\eta_{-1} = \eta_0 = \hat{z}_0 - z_0$, and setting $\hat{z}_0 - z_0 = \hat{y}_0 - y_0$, one can derive the two-step recurrence

$$\eta_{i+1} = \mathrm{diag}(\epsilon_{\pi,i})\big(I-\hat{H}_n\big)\big(\eta_i + \hat{\beta}(\eta_i - \eta_{i-1}) + \epsilon_{y,i-1}\big) + \epsilon_{t,i}$$

for the accumulated arithmetic error at each iteration. Note that the error η_i at each iteration is inherently bounded by the box $\widehat{\mathbb{K}}$. However, in the absence of the projection operation of line 3 and the associated error truncation, these errors remain bounded. To show this, we can express the evolution of the arithmetic error using the two-step recurrence

$$\begin{aligned}\underbrace{\begin{bmatrix}\eta_{i+1}\\ \eta_i\end{bmatrix}}_{=:\xi_{i+1}} &= \underbrace{\begin{bmatrix}\big(1+\hat{\beta}\big)\big(I-\hat{H}_n\big) & -\hat{\beta}\big(I-\hat{H}_n\big)\\ I & 0\end{bmatrix}}_{=:A}\underbrace{\begin{bmatrix}\eta_i\\ \eta_{i-1}\end{bmatrix}}_{\xi_i}\\ &\quad + \underbrace{\begin{bmatrix}\big(I-\hat{H}_n\big) & I\\ 0 & 0\end{bmatrix}}_{=:B}\underbrace{\begin{bmatrix}\epsilon_{y,i-1}\\ \epsilon_{t,i}\end{bmatrix}}_{=:\upsilon_i},\end{aligned} \tag{15}$$

and then show that this linear system is stable. Recalling Assumption 1, which bounds the eigenvalues of $\hat{H}_n$ in the interval $(0, 1]$ and $\hat{\beta}$ in the interval $[0, 1)$, we can use the following result:

Lemma 1. *Let C be any symmetric positive definite matrix with maximum eigenvalue less than or equal to one. For every constant γ in the interval $[0,1]$ the matrix*

$$\mathcal{M} = \begin{bmatrix} (1+\gamma)(I-C) & -\gamma(I-C) \\ I & 0 \end{bmatrix}$$

is Schur stable, i.e. its spectral radius $\rho(\mathcal{M})$ is less than one.

Proof. Assume the eigenvalue decomposition $I-C = V^T \Lambda V$, with Λ diagonal with entries $\lambda_i \in [0,1)$. The eigenvalues of $\mathcal{M}$ are unchanged by left- and right-multiplication by $\left[\begin{smallmatrix} V & \\ & V \end{smallmatrix}\right]$ and its transpose. It is therefore sufficient to examine instead the spectral radius of

$$\mathcal{M}_D = \begin{bmatrix} (1+\gamma)\Lambda & -\gamma\Lambda \\ I & 0 \end{bmatrix}.$$

Since this matrix has exclusively diagonal blocks, its eigenvalues coincide with those of the two-by-two submatrices

$$\mathcal{M}_{D,i} = \begin{bmatrix} (1+\gamma)\lambda_i & -\gamma\lambda_i \\ 1 & 0 \end{bmatrix}, \quad \text{for } i = 1,\dots,n,$$

and it is sufficient to prove that every such submatrix has spectral radius less than one. Note that the eigenvalues of $\mathcal{M}_{D,i}$ are the roots of the characteristic equation

$$\mu^2 - (1+\gamma)\lambda_i \mu + \lambda_i \gamma = 0. \tag{16}$$

It is easily verified that a sufficient condition for any quadratic equation in the form

$$x^2 + 2bx + c = 0$$

to have roots strictly inside the unit disk is for its coefficients to satisfy i) $|b| < 1$, ii) $c < 1$ and iii) $2|b| < c+1$. For the eigenvalue solutions to (16), this amounts to i) $(1+\gamma)\lambda_i/2 < 1$, ii) $\lambda_i\gamma < 1$ and iii) $(1+\gamma)\lambda_i < \gamma\lambda_i + 1$. All three conditions are easily confirmed for the case $\lambda_i \in [0,1)$, $\gamma \in [0,1]$. □

2) Stability of Arithmetic Errors in ADMM:

As in the preceding section, for ADMM one can analyze in detail the arithmetic error introduced at each step of a fixed-point implementation of Algorithm 2.

Defining $\eta_i := \hat{z}_i - z_i$, $\gamma_i := \hat{\nu}_i - \nu_i$, a similar analysis to that of the preceding section produces the two-step error recurrence

$$\underbrace{\begin{bmatrix} \eta_{i+1} \\ \gamma_{i+1} \end{bmatrix}}_{=:\xi_{i+1}} = \underbrace{\begin{bmatrix} \rho\,\mathrm{diag}(\epsilon_{\pi,i})\,\hat{M}_{11} & -\mathrm{diag}(\epsilon_{\pi,i})\,(\hat{M}_{11} - \frac{1}{\rho}I) \\ \rho^2 \hat{M}_{11}(I - \mathrm{diag}(\epsilon_{\pi,i})) & (I - \rho\hat{M}_{11})(I - \mathrm{diag}(\epsilon_{\pi,i})) \end{bmatrix}}_{=:A} \underbrace{\begin{bmatrix} \eta_i \\ \gamma_i \end{bmatrix}}_{\xi_i} + \underbrace{\begin{bmatrix} \mathrm{diag}(\epsilon_{\pi,i}) & 0 \\ \rho(I - \mathrm{diag}(\epsilon_{\pi,i})) & I \end{bmatrix}}_{=:B} \underbrace{\begin{bmatrix} \epsilon_{y,i} \\ \epsilon_{\nu,i} \end{bmatrix}}_{=:\upsilon_i}, \tag{17}$$

where: $\epsilon_{y,i} \in [-n2^{-b}, 0]^n$ is a vector of multiplication errors arising from Algorithm 2, line 2; $\epsilon_{\pi,i} \in [0,1]^n$ is a vector of error reduction scalings arising from the projection operation in line 3; and $\epsilon_{\nu,i} \in [-2^{-b}, 2^{-b}]^n$ is a vector of multiplication errors arising from line 4 with $\epsilon_{\nu,-1} = 0$. Note that one can show that even when $\widehat{\mathbb{K}}$ is not a box in the presence of soft state constraints, the error can only be reduced by the projection operation. The initial iterates of the recurrence relation are $\eta_{-1} = \eta_0$, where $\eta_0 := \hat{z}_0 - z_0$.

As in the case of the fast gradient method, the arithmetic error η_i is inherently bounded by the constraint set $\widehat{\mathbb{K}}$. However, even in the absence of these bounding constraints (so that $\mathrm{diag}(\epsilon_{\pi,i}) = I$), one can still establish that the arithmetic errors are bounded via examination of the eigenvalues of the matrix

$$\mathcal{N} := \begin{bmatrix} \rho\hat{M}_{11} & -(\hat{M}_{11} - \frac{1}{\rho}I) \\ 0 & 0 \end{bmatrix}. \tag{18}$$

Recalling Assumption 2, we have the following result:

Lemma 2. *The matrix N in* (18) *is Schur stable for any $\rho > 0$.*

Proof. The eigenvalues of (18) are either 0 or $\rho\lambda_i(\hat{M}_{11})$, so it is sufficient to show that the symmetric matrix $\hat{M}_{11}$ satisfies $\rho\|\hat{M}_{11}\| < 1$. Recalling that

$$\begin{bmatrix} \hat{M}_{11} & \hat{M}_{12} \\ \hat{M}_{12}^T & \hat{M}_{22} \end{bmatrix} = \begin{bmatrix} \hat{Z} & \hat{F}^T \\ \hat{F} & 0 \end{bmatrix}^{-1}$$

where $\hat{Z} := \hat{H}_A + \rho I \succ 0$, the matrix inversion lemma provides the identity

$$\begin{aligned} \hat{M}_{11} &= \hat{Z}^{-\frac{1}{2}} \left[I - \hat{Z}^{-\frac{1}{2}} \hat{F}^T (\hat{F}\hat{Z}^{-1}\hat{F}^T)^{-1} \hat{F}\hat{Z}^{-\frac{1}{2}} \right] \hat{Z}^{-\frac{1}{2}} \\ &=: \hat{Z}^{-\frac{1}{2}} \hat{P} \hat{Z}^{-\frac{1}{2}}, \end{aligned} \tag{19}$$

where $\hat{P}$ is a projection onto the kernel of $\hat{F}\hat{Z}^{-\frac{1}{2}}$, hence $\|\hat{M}_{11}\| \le \|\hat{Z}^{-\frac{1}{2}}\|\|\hat{P}\|\|\hat{Z}^{-\frac{1}{2}}\| = \|\hat{Z}^{-1}\|$. It follows that

$$\rho\|\hat{M}_{11}\| \le \rho\|(\hat{H}_A + \rho I)^{-1}\| = \rho \cdot \frac{1}{\lambda_{\min}(\hat{H}_A) + \rho} \le 1,$$

where $\lambda_{\min}(\hat{H}_A)$ is the smallest eigenvalue of the positive semidefinite matrix $\hat{H}_A$. If $\hat{H}_A$ is actually positive definite, then the preceding inequality is strict and the proof is complete.

Otherwise, to prove that the inequality is strict we must show that $1/\rho$ is not an eigenvalue for $\hat{M}_{11}$ (which is positive semidefinite by virtue of (19)). Assume the contrary, so that there exists some eigenvector v of $\hat{M}_{11}$ with eigenvalue $1/\rho$, and some additional (arbitrary) vector q that solves the linear system

$$\begin{bmatrix} v \\ q \end{bmatrix} = \begin{bmatrix} \hat{Z} & \hat{F}^T \\ \hat{F} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \rho \cdot v \\ 0 \end{bmatrix}.$$

Any solution must then satisfy both $\hat{H}_A v \in \mathrm{Im}(\hat{F}^T)$ and $v \in \mathrm{Ker}(\hat{F})$. Consequently $v^T \hat{H}_A v = 0$, which requires $v \in \mathrm{Ker}(\hat{H}_A)$ since $\hat{H}_A$ is positive semidefinite. Recall that any such v can be decomposed into $v = (u_0, \dots, u_{N-1}, x_0, \delta_0, \dots, x_N, \delta_N)$. If the quadratic penalty for each δ_i is positive definite, then $v \in \mathrm{Ker}(\hat{H}_A)$ requires each $\delta_i = 0$.

Since $\hat{F}v = 0$, the remaining components of v must correspond to a sequence of state and inputs compatible with the system dynamics in (2), starting from an initial state $x_0 = 0$. Any solution $v \neq 0$ would then require at least one component $u_i \neq 0$. Then $v^T \hat{H}_A v \ge u_i^T R u_i > 0$ since R is assumed positive definite, a contradiction.

□

3) Arithmetic Errors Bounds for the Fast Gradient Method and ADMM:

Finally, for both the fast gradient method and ADMM we can use Lemmas 1 and 2 to establish an upper bound on the magnitude of error η_i for *any* arithmetic round-off errors that might have occurred up to iteration i.

Proposition 2. *Let b be the number of fraction bits and n be the dimension of the decision vector. Consider the error dynamics due to arithmetic round-off in* (15) *or in (17), assuming no error reduction from projection. The magnitude of* any *accumulation of round-off errors up to iteration i, $\|\eta_i\| = \|\hat{z}_i - z_i\|$, is upper-bounded by*

$$\bar{\eta}_i = \|EA^i\| \left\| \begin{bmatrix} \eta_0 \\ \eta_0 \end{bmatrix} \right\| + 2^{-b}\sqrt{n(1+n^2)} \sum_{k=0}^{i-1} \|EA^{i-1-k}B\| \quad (20)$$

for all $i = 0, \ldots, I_{\max} - 1$, where matrix $E = \begin{bmatrix} I & 0 \end{bmatrix}$.

Proof. From the one-step recurrence (15) or (17) we find that

$$\xi_i = A^i \xi_0 + \sum_{k=0}^{i-1} A^{i-1-k} B \upsilon_k, \quad i = 0, 1, \ldots I_{\max} - 1,$$

such that the result is obtained from applying the properties of the matrix norm. Observe that $2^{-b}\sqrt{n(1+n^2)}$ is the maximum magnitude of υ_k for any $k = 0, \ldots, i-1$. □

Since the matrix A is Schur stable, the bound in (20) converges. Indeed, the effect of the initial error ξ_0 decays according to

$$\|EA^i\| \propto \max_j |\lambda_j|^i, \quad (21)$$

whereas the term driven by arithmetic round-off errors in every iteration behaves according to

$$\sum_{k=0}^{i-1} \|EA^{i-1-k}B\| \propto \frac{1}{1 - \max_j |\lambda_j|} - \frac{\max_j |\lambda_j|^i}{1 - \max_j |\lambda_j|}. \quad (22)$$

This result can be used to choose the number of bits b *a priori* to meet accuracy specifications on the minimizer, as illustrated with an example in Figure 6.

V. Embedded Hardware Architectures for First-Order Solution Methods

Amdahl's law [38] states that the potential acceleration of an optimization algorithm through parallelization is limited by the fraction of sequential dependencies in the algorithm. First-order optimization methods such as the fast gradient method and ADMM have a smaller number of sequential dependencies than interior-point or active-set methods. In fact, a very large fraction of the computation involves a single readily parallelizable matrix-vector multiplication, hence the expected benefit from parallelization is substantial. Our implementations of both the fast gradient method (Algorithm 1) and ADMM (Algorithm 2) differ somewhat from more conventional implementations of these methods in order to minimize sequential dependencies. Observe that in both of our algorithms, the computations of the individual vector components are independent and the only communication occurs during matrix-vector multiplication. This allows for efficient parallelization

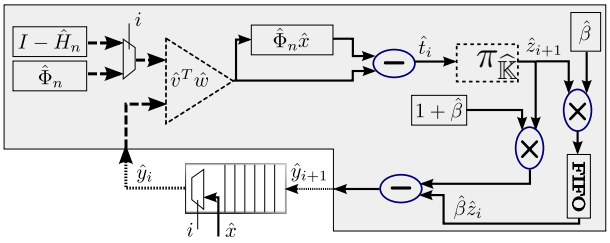


Fig. 1: Fast gradient compute architecture. Boxes denote storage elements and dotted lines represent Nn_u parallel vector links. The dot-product block $\hat{v}^T\hat{w}$ and the projection block $\pi_{\widehat{\mathbb{K}}}$ are depicted in Figures 2a and 3a in detail. FIFO stands for first-in first-out memory and is used to hold the values of the current iterate for use in the next iteration. In the initial iteration, the multiplexers allow $\hat{x}$ and $\hat{\Phi}_n$ through and the result $\hat{\Phi}_n\hat{x}$ is stored in memory. In the subsequent iterations, the multiplexers allow $\hat{y}_i$ and $I - \hat{H}_n$ through and $\hat{\Phi}_n\hat{x}$ is read from memory.

given the custom computing and communication architectures discussed next. Specifically, we describe a tool that takes as inputs the data type, number of bits, level of parallelism and the delays of an adder/subtracter (l_A) and multiplier (l_M) and automatically generates a digital architecture described in the VHDL hardware description language.

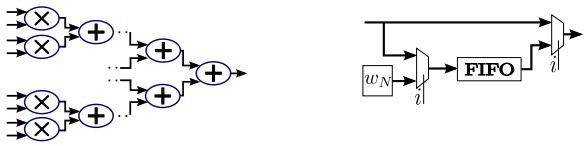
A. Hardware Architecture for the Fast Gradient Method

For a fixed-point data type, the parameterized architecture implementing Algorithm 1 for problem (6) is depicted in Figure 1. The matrix-vector multiplication is computed in the block labeled $\hat{v}^T\hat{w}$, which is shown in detail in Figure 2a. It consists of an array of Nn_u parallel multipliers followed by an adder reduction tree of depth $\lceil \log_2 Nn_u \rceil$. The architecture for performing the projection operation on the set $\widehat{\mathbb{K}}$ is shown in Figure 3a. It compares the incoming value with the upper and lower bounds for that component. Based on the result, the component is either saturated or left unchanged.

The amount of parallelism in the circuit is parameterized by the parameter P. In Figure 1, $P = 1$, meaning that there is parallelism within each dot-product but the Nn_u dot-products required for matrix-vector multiplication are computed sequentially. If the level of parallelization is increased to $P = 2$, there will be two copies of the shaded circuit in Figure 1 operating in parallel: with one unit computing the odd components of $\hat{y}_i$ and $\hat{z}_i$ and with the other unit computing the even components. The different blocks communicate through a serial-to-parallel shift register that accepts P serial streams and outputs Nn_u parallel values for matrix-vector multiplication. These Nn_u values are the same for all blocks. It takes $\lceil \frac{Nn_u}{P} \rceil$ clock cycles to have enough data to start a new iteration, hence the number of clock cycles needed to compute one iteration of the fast gradient method for $P \in \{1, \ldots, Nn_u\}$ is

$$L_F := \left\lceil \frac{Nn_u}{P} \right\rceil + l_A \lceil \log_2 Nn_u \rceil + 2l_M + 3l_A + 1\,. \quad (23)$$

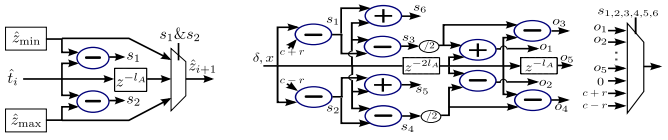
Expression (23) suggests that there will be diminishing returns to parallelization – a consequence of Amdahl's law.



(a) Dot-product block with parallel tree architecture.

(b) Hardware support for warm-starting, which adds one cycle delay. The last entries of the vector are padded with w_N, which can be constant or depend on previous values.

Fig. 2: Architectures of dot-product and warm-starting.



(a) Box projection block. The total delay from $\hat{t}_i$ to $\hat{z}_{i+1}$ is $l_A + 1$.

(b) Cone projection block. The total delay for each component is $2l_A + 1$. x and δ are assumed to arive and leave in sequence.

Fig. 3: Projection architectures. A delay of l_A cycles is denoted by z^{-l_A}.

However, (23) also suggests that if there are enough resources available, the effect of the problem size on increased computational delay is only logarithmic in the worst case. As Moore's law continues to deliver devices with greater transistor densities, the possibility of implementing algorithms in a fully parallel fashion for medium size optimization problems is becoming a reality.

B. Hardware Architecture for ADMM

Algorithm 2 shares the same computational patterns with Algorithm 1. Matrices $\hat{M}_{11}$ and $\hat{M}_{12}$ have the same dense structure as matrices $I - \hat{H}_n$ and $\hat{\Phi}_n$, hence the high-level architecture is very similar and we do not include it here to avoid replication. The differences lie in the size of the matrices, which affect the number of clock cycles to compute one iteration

$$L_A := \left\lceil \frac{n_A}{P} \right\rceil + l_A \lceil \log_2 (n_A) \rceil + l_M + 6 l_A + 2 \,, \qquad (24)$$

where $n_A := N(n_u + n_x + |\mathcal{S}|) + n_x + |\mathcal{S}|$, warm-starting support for variables z and ν (shown in Figure 2b), and the projection block for supporting soft state constraints described in Figure 3b. This block performs the projection of the pair (x, δ) onto the set satisfying $\{|x - c| \leq r + \delta, \delta \geq 0\}$ by using an explicit solution map for the projection operation and computing the search procedure efficiently. In fact, only l_A extra cycles are needed compared to the standard hard-constrained projection. The block performs a set of comparisons that are used to drive the select signal of a multiplexer.

Note that since multiplication and division by powers of two requires no resources in hardware (just a reinterpretation of an array of signals), if ρ is restricted to be a power of two, no hardware multipliers are required in ADMM outside of the matrix-vector multiplication block. Table I compares the resources required to implement the two architectures. Again, with ADMM we trade higher resource requirements and longer delays for the capability to solve more general problems.

TABLE I: Resources required for the fast gradient and ADMM computing architectures.

	Fast gradient	ADMM
multipliers	$P\,[Nn_u + 2]$	Pn_A
adders/subtracters	$P\,[Nn_u + 3]$	$P\,[n_A + 15]$
memory blocks	$P\,[Nn_u + n_x + 4]$	$P\,[n_A + 8]$
size of memory blocks	$\left\lceil \frac{Nn_u}{P} \right\rceil$	$\left\lceil \frac{n_A}{P} \right\rceil$

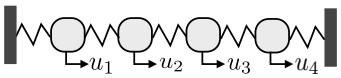


Fig. 4: Oscillating masses example.

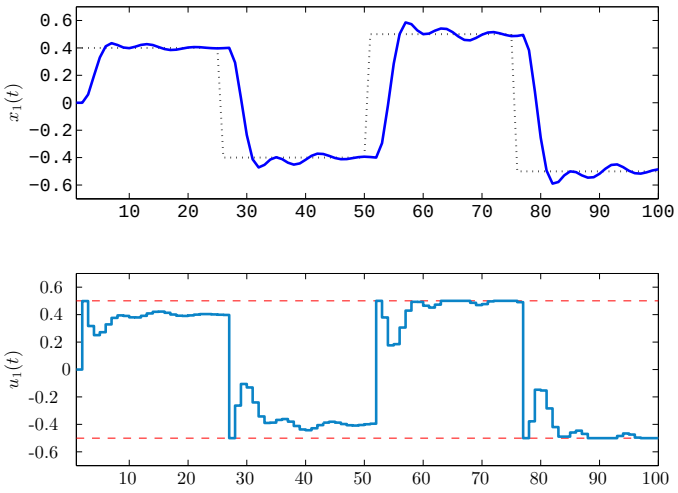
Note that in a custom hardware implementation of either of our two methods, the number of execution cycles per iteration is exact. We also employ a fixed number of iterations in our implementations of both algorithms, rather than implementing a numerical convergence test, since such convergence tests represent a somewhat self-defeating computational bottleneck in a hard real-time context. Providing cycle accurate completion guarantees is critical for reliability in high-speed real-time applications [39].

VI. Numerical Benchmark Study

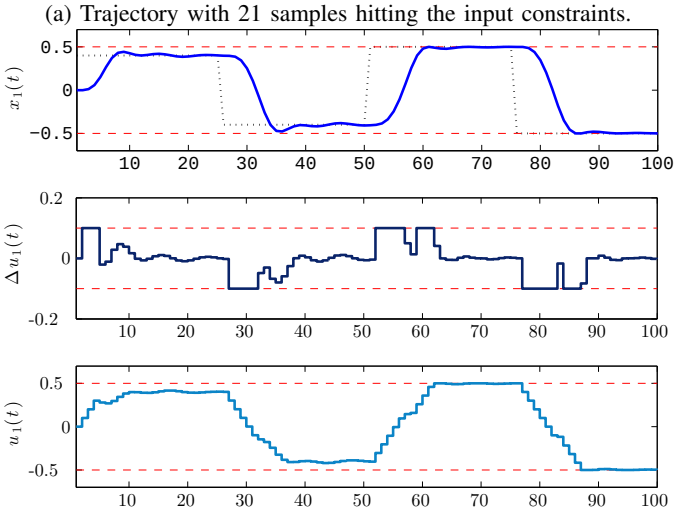
We reported an implementation of the fast gradient architecture in the preliminary publication [19] to implement an input-constrained MPC controller for a real-world, highly dynamic positioning system inside an atomic force microscope requiring a sampling rate in excess of 1MHz. In this paper, for easier comparison with the existing literature, we use a widely studied benchmark example consisting of a set of oscillating masses attached to walls [10], [40], as illustrated by Figure 4. The system is sampled every 0.5 seconds assuming a zero-order hold and the masses and the spring constants have a value of 1kg and 1Nm^{-1}, respectively[3]. The system has four control inputs and two states for each mass, its position and velocity, for a total of eight states. The goal of the controller, with parameters $N = 10$, $Q = I$ and $R = I$, is to track a reference for the position of each mass while satisfying the system limits.

We consider first the case where the control inputs are constrained to the interval $[-0.5, 0.5]$ and the optimization problem (6) with 40 optimization variables is solved via the fast gradient method. Secondly, we consider additional hard constraints on the rate of change in the inputs on the interval $[-0.1, 0.1]$ and soft constraints on the states corresponding to the mass positions on the interval $[-0.5, 0.5]$. The remaining states are left unconstrained. The state is augmented to enforce input-rate constraints, and the further inclusion of slack variables increases the dimension of the state vector to $n_x = 12$. Note that for problems of this size, MPC control designs

[3]Note that we choose this sampling time and parameter set for ease of comparison to other published results. Our implemented methods require computation times on the order of 1μs, as we report later in this section.



(a) Trajectory with 21 samples hitting the input constraints.



(b) Trajectory with 11, 28 and 14 samples hitting the input, rate and output constraints, respectively.

Fig. 5: Closed-loop trajectories showing actuator limits, desirable output limits and a time-varying reference. MPC allows for optimal operation on the constraints.

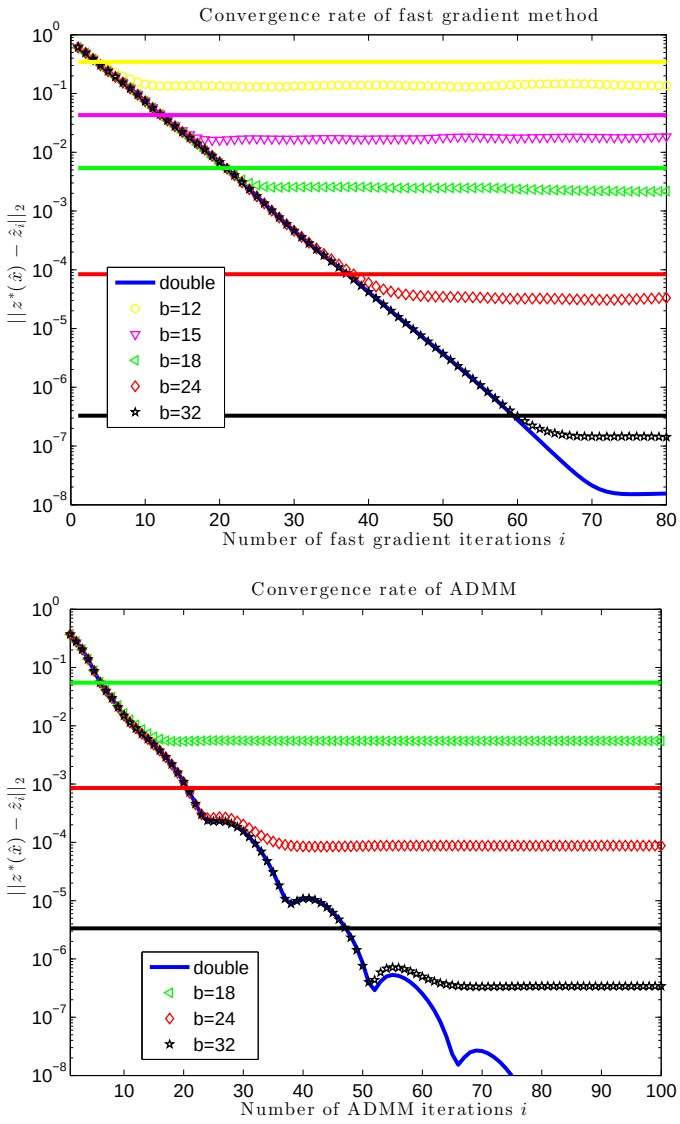


Fig. 6: Practical convergence behavior of the fast gradient method (top) and ADMM (bottom) under different number representations. Theoretical error bounds given by (20) are represented by solid lines.

based on parametric programming [3], [4] are generally not tenable, necessitating online optimization methods. The resulting problem with 216 optimization variables in the form (9) is solved via ADMM. The closed-loop trajectories using an MPC controller based on a double precision solver running to optimality are shown in Figure 5, where all the constraints become active for a significant portion of the simulation. We do not include any disturbance model in our simulation, although the presence of an exogenous disturbance signal would not lead to infeasibility since the MPC implementation includes only soft-constrained states. Trajectories arising from closed-loop simulation using a controller based on our fixed-point methods are indistinguishable from those in Figure 5, so are excluded for brevity.

As a reference for later comparison, an input-constrained problem with two inputs and 10 states, formulated as an optimization problem of the form (6) with 40 variables, was solved in [40] using the fast gradient method in approximately 50 μseconds. In terms of state-constrained implementations, a problem with three inputs and 12 states, formulated as a sparse quadratic program with hard state constraints and 300 variables, was solved in [10] using an interior-point method reporting computing times in the region of 5 milliseconds, while the state constraints remained inactive. In both cases, the solvers were implemented in software on high-performance desktop machines.

Our goal is to choose the minimum number of bits and solver iterations such that the closed-loop performance is satisfactory while minimizing the amount of resources needed to achieve certain sampling frequencies. Assumptions 1 and 2 impose a lower bound on the number of bits given by $b \geq 10$ and $b \geq 18$ for the input- and state-constrained problems, respectively. Figure 6 shows the convergence behavior of the fast gradient method and ADMM for two samples in

TABLE II: Percentage difference in average closed-loop cost with respect to a standard double precision implementation. In each table, b is the number of fraction bits employed and $I_{\max}$ is the (fixed) number of algorithm iterations. In certain cases, the error increases with the number of iterations due to increasing, yet bounded, accumulation of round-off errors.

$I_{\max}\backslash b$	10	12	14	16	18	20
5	5.30	2.76	2.87	3.03	3.05	3.06
10	14.53	0.14	0.06	0.18	0.20	0.02
15	17.04	0.35	0.25	0.04	0.00	0.01
20	16.08	0.15	0.19	0.06	0.01	0.00
25	17.27	0.15	0.19	0.05	0.01	0.00
30	16.90	0.31	0.21	0.03	0.02	0.00
35	18.44	0.19	0.22	0.05	0.01	0.00

(a) FGM

$I_{\max}\backslash b$	10	12	14	16	18	20
10	53.49	0.18	1.17	0.68	0.57	0.58
15	47.84	0.46	1.08	0.63	0.51	0.49
20	44.79	0.76	0.95	0.57	0.45	0.42
25	47.03	0.98	0.86	0.51	0.39	0.37
30	45.17	1.02	0.82	0.46	0.35	0.32
35	46.02	1.07	0.81	0.43	0.31	0.28
40	46.87	1.29	0.74	0.41	0.28	0.25

(b) ADMM

the simulation with an actively constrained solution. The theoretical error bounds on the residual round-off error η_i, given by (20), allow one to make practical predictions for the actual error for a given number of bits, which, as predicted by Lemma 2 and (21) and (22), converges to a finite value. Table II shows the relative difference in closed-loop tracking performance for different fixed-point fast gradient and ADMM controllers compared to the optimal controller. Assuming that a relative error smaller than 0.05% is desirable, using 15 solver iterations and 16 fraction bits would be a suitable choice for the fast gradient method. The problem (9) solved via ADMM appears more vulnerable to reduced precision implementation, although satisfactory control performance can still be achieved using a surprisingly small number of bits. In this case, employing more than 18 fraction bits or more than 40 ADMM iterations results in insignificant improvements.

For the implementation of ADMM there are a number of tuning parameters left to the control designer. Setting the regularization parameter to $\rho = 2$ simplifies the implementation and provided good convergence behavior. The maximum observed value for the Lagrange multipliers ν was 7.8, so the penalty parameter σ_1 was set to $\sigma_1 = 8$ to obtain an exact penalty formulation as described by Theorem 1. In Section III-C it was noted that the convergence of ADMM can be very slow when there is large mismatch between the size of the primal and dual variables. This problem can be largely avoided by scaling the matching condition (10) with a diagonal matrix, where the entries associated with the soft-constrained states and the slack variables are assigned σ and the rest are assigned 1. This scaling procedure correspond to variable transformations $y = D\tilde{y}$ and $z = D\tilde{z}$ that can be applied offline.

In order to evaluate the potential computing performance the architectures described in Section V were implemented in FPGAs. For a fixed number of iterations one can calculate the execution time of the solver deterministically according to (23) or (24). The FPGA designs can be clocked at more than 400 MHz using chips from Xilinx's high-performance Virtex 6 family or at more than 230 MHz using devices from the low cost and low power Spartan 6 family. Table III shows the achievable sampling times on the two families for different levels of parallelization. The resource usage is stated in terms of the number of embedded multiplier blocks since this is the limiting resource in these designs. For the input-constrained problem solved via the fast gradient method, one can achieve sampling rates beyond 1 MHz with Virtex 6 devices using a modest amount of parallelization. One can also achieve sampling rates in the region of 700 kHz with Spartan 6 devices consuming in the region of 1 W of power. For the state-constrained problem solved via ADMM, since the number of variables is significantly larger, larger devices are needed and longer computational times have to be tolerated. In this case, achievable solution times range from 40kHz to 200kHz for different Virtex 6 devices.

Note that the fastest performance numbers reported in the literature are in the millisecond region, several orders of magnitude slower than what is achievable using the techniques presented in this paper.

TABLE III: Resource usage and potential performance at 400MHz (Virtex6) and 230MHz (Spartan6) with 15 and 40 solver iterations for FGM (Table IIIa) and ADMM (Table IIIb), respectively. The suggested chips in the bottom two rows of each table are the smallest with enough embedded multipliers to support the resource requirements of each implementation.

P	1	2	3	4	8	16	32
multipliers	42	84	126	168	336	672	1344
V6 T_s (μs)	1.95	1.20	0.98	0.82	0.64	0.56	0.53
S6 T_s (μs)	3.39	2.09	1.70	1.43	1.10	0.98	0.91
V6 chip	LX75	LX75	LX75	LX75	LX130	LX240	SX315
S6 chip	LX45	LX75	LX75	LX100	-	-	-

(a) FGM

P	1	2	3	4	5	6	7
multipliers	216	432	648	864	1080	1296	1512
V6 T_s (μs)	23.40	12.60	9.00	7.20	6.20	5.40	4.90
S6 T_s (μs)	40.70	21.91	15.65	12.52	10.78	9.39	8.52
V6 chip	LX75	LX130	LX240	LX550	SX315	SX315	SX475
S6 chip	-	-	-	-	-	-	-

(b) ADMM

VII. Acknowledgements

This work was supported by the EPSRC (Grants EP/G031576/1 and EP/I012036/1) and the EU FP7 Project EMBOCON, as well as industrial support from Xilinx, the Mathworks, and the European Space Agency.

References

[1] J. M. Maciejowski, *Predictive Control with Constraints*. Harlow, UK: Pearson Education, 2001.

[2] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Publishing, 2009.

[3] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan 2002.

[4] F. Comaschi, B. A. G. Genuit, A. Oliveri, W. P. Heemels, and M. Storace, "FPGA implementations of piecewise affine functions based on multi-resolution hyperrectangular partitions," *IEEE Transactions on Circuits and Systems I*, vol. 59, no. 12, pp. 2920–2933, Dec 2012.

[5] S. Richter, C. Jones, and M. Morari, "Computational complexity certification for real-time MPC with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, Jun 2012.

[6] S. Richter, M. Morari, and C. Jones, "Towards computational complexity certification for constrained MPC based on lagrange relaxation and the fast gradient method," in *Proc. 50th IEEE Conf. on Decision and Control*, Orlando, USA, Dec 2011, pp. 5223–5229.

[7] M. Kögel and R. Findeisen, "Parallel solutions of model predictive control using the alternating direction method of multipliers," in *Proc. 4th IFAC Conf. on Nonlinear Model Predictive Control*, Noordwijkerhout, Netherlands, 2012, pp. 369–374.

[8] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *Proc. 51st IEEE Conf. on Decision and Control*, Maui, HI, USA, Dec 2012.

[9] M. Annergren, A. Hansson, and B. Wahlberg, "An ADMM algorithm for solving l_1 regularized MPC," in *Proc. 51st IEEE Conf. on Decision and Control*, Maui, HI, USA, Dec 2012.

[10] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, Mar 2010.

[11] A. Domahidi, A. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient interior point methods for multistage problems arising in receding horizon control," in *Proc. 51th IEEE Conf. on Decision and Control*, Maui, HI, USA, Dec 2012.

[12] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control: Automatic generation of high-speed solvers," *IEEE Control Systems Magazine*, vol. 3, no. 31, pp. 52–65, 2011.

[13] F. Ullmann, "FiOrdOs: A Matlab toolbox for C-code generation for first order methods," Master's thesis, ETH Zürich, 2011. [Online]. Available: fiordos.ethz.ch/

[14] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, Jul 2008.

[15] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "An FPGA implementation of a sparse quadratic programming solver for constrained predictive control," in *Proc. ACM Symp. Field Programmable Gate Arrays*, Monterey, CA, USA, Mar 2011.

[16] P. D. Vouzis, L. G. Bleris, M. G. Arnold, and M. V. Kothare, "A system-on-a-chip implementation for embedded real-time model predictive control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1006–1017, Sep 2009.

[17] A. G. Wills, G. Knagge, and B. Ninness, "Fast linear model predictive control via custom integrated circuit architecture," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 59–71, 2012.

[18] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "Towards a fixed-point QP solver for predictive control," in *Proc. 51th IEEE Conf. on Decision and Control*, Maui, HI, USA, Dec 2012.

[19] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded predictive control on an FPGA using the fast gradient method," in *Proc. European Control Conf.*, Zürich, Switzerland, Jul 2013, p. (submitted).

[20] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000.

[21] M. N. Zeilinger, C. N. Jones, and M. Morari, "Robust stability properties of soft constrained MPC," in *Proc. 49th IEEE Conf. on Decision and Control*, Atlanta, GA, USA, Dec 2010, pp. 5276–5282.

[22] P. O. M. Scokaert and J. B. Rawlings, "Feasibility issues in linear model predictive control, feasibility issues in linear model predictive control," *AIChE Journal, AIChE Journal*, vol. 45, no. 8, pp. 1649–1659, Aug. 1999.

[23] L. McGovern and E. Feron, "Closed-loop stability of systems driven by real-time, dynamic optimization algorithms," in *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, AZ, Dec 1999, pp. 3690–3696.

[24] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, Massachusetts: Athena Scientific, 1999.

[25] Y. Nesterov, "A method for solving a convex programming problem with convergence rate $1/k^2$," *Soviet Math. Dokl.*, vol. 27, no. 2, pp. 372–376, 1983.

[26] ——, *Introductory Lectures on Convex Optimization. A Basic Course*. Springer, 2004.

[27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[28] B. O'Donoghue, G. Stathopoulos, and S. Boyd, "A splitting method for optimal control," *IEEE Transactions on Control Systems Technology*, 2013 (to appear).

[29] D. Boley, "Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2183–2207, Nov 2013.

[30] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems," *arXiv:1306.2454*, June 2013. [Online]. Available: http://arxiv.org/abs/1306.2454

[31] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Jan. 1997.

[32] H. Bingsheng and Y. Xiaoming, "On the $O(1/t)$ convergence rate of alternating direction method," Nanjing University, China, Tech. Rep., Oct. 2011. [Online]. Available: http://www.optimization-online.org/DB_HTML/2011/09/3157.html

[33] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," Rice University, Tech. Rep., Oct. 2012. [Online]. Available: http://www.caam.rice.edu/ wy1/papers/genenarlized_admm_linear_conv.html

[34] (2011) LogiCORE IP floating-point operator v5.0. Xilinx. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/floating_point_ds335.pdf

[35] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, 1st ed., ser. Notes on Applied Science. London, UK: Her Majesty's Stationary Office, 1963, no. 32.

[36] M. Baes, "Estimate sequence methods: Extensions and approximations," Zurich, Nov. 2009.

[37] M. Schmidt, N. L. Roux, and F. Bach, "Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization," *arXiv:1109.2415*, Sept. 2011. [Online]. Available: http://arxiv.org/abs/1109.2415

[38] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc. AFIPS Joint Computer Conference*, Atlantic City, NJ, USA, Apr 1967, pp. 483–485.

[39] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 1st ed., 2011. [Online]. Available: http://LeeSeshia.org

[40] M. Kögel and R. Findeisen, "A fast gradient method for embedded linear predictive control," in *Proc. 18th IFAC World Congress*, Milano, Italy, Aug 2011.

Juan L. Jerez (S'11) received the M.Eng degree in electrical engineering from Imperial College London, UK in 2009 and a Ph.D from the Circuits and Systems research group at the same institution in 2013. He is currently a postdoctoral researcher at ETH Zürich and founder of embotech GmbH. The focus of his research work is on developing tailored linear algebra and optimization algorithms for efficient implementation on custom parallel computing platforms and embedded systems. Dr Jerez has been awarded the IET Doctoral Dissertation Prize in Control and Automation and a Pioneer Fellowship from ETH Zürich.

Stefan Richter received an MSc in Telematics from the Technical University of Graz, Austria, in 2007. In 2012 he obtained a PhD from ETH Zürich for his work on computational complexity certification of first-order methods for model predictive control. He is currently a Postdoc at the Automatic Control Laboratory at ETH Zürich where his research focuses on convex optimization in the context of control systems.

George A. Constantinides (S'96-M'01-SM'08) received the Ph.D. degree from Imperial College London in 2001. Since 2002, he has been with the faculty at Imperial College London, where he is currently Professor of Digital Computation and Head of the Circuits and Systems research group. He will be program (general) chair of the ACM International Symposium on Field-Programmable Gate Arrays in 2014 (2015). He serves on several programme committees and has published over 150 research papers in peer refereed journals and international conferences. Dr Constantinides is a Senior Member of the IEEE and a Fellow of the British Computer Society.

Eric C. Kerrigan (S'94-M'02) received a PhD from the University of Cambridge in 2001 and has been with Imperial College London since 2006. His research is focused on the development of efficient numerical methods and embedded computing architectures for solving advanced optimization, control and estimation problems in real-time. His work is applied to a variety of problems in aerospace and renewable energy applications. He is on the IEEE Control Systems Society Conference Editorial Board and is an associate editor of the IEEE Transactions on Control Systems Technology, Control Engineering Practice, and Optimal Control Applications and Methods.

Paul Goulart received the S.B. and S.M. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. degree in 2007 from the University of Cambridge, Cambridge, U.K., where he was a Gates Cambridge Scholar. From 2007 to 2011, he was a Lecturer in control systems in the Department of Aeronautics, Imperial College London, and is currently with the Automatic Control Laboratory, ETH Zurich. His research interests include robust and predictive control, robust optimization, and control of fluid flows.

Manfred Morari (F05) received the Diploma degree from ETH Zürich, Zürich, Switzerland, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, both in chemical engineering. He was appointed Head of the Department of Information Technology and Electrical Engineering, ETH Zrich, in 2009. He was Head of the Automatic Control Laboratory from 1994 to 2008. Before that, he was the McCollum-Corcoran Professor of Chemical Engineering and Executive Officer for Control and Dynamical Systems, California Institute of Technology, Pasadena, CA, USA. His interests are in hybrid systems and the control of biomedical systems. He has held appointments with Exxon and ICI plc and serves on the technical advisory boards of several major corporations. Dr. Morari has received numerous awards in recognition of his research contributions, among them the Donald P. Eckman Award, the John R. Ragazzini Award and the Richard E. Bellman Control Heritage Award of the American Automatic Control Council, the Allan P. Colburn Award and the Professional Progress Award of the AIChE, the Curtis W. McGraw Research Award of the ASEE, Doctor Honoris Causa from Babes-Bolyai University, IFAC and AIChE, and the IEEE Control Systems Technical Field Award. He