2013 European Control Conference (ECC)
July 17-19, 2013, Zürich, Switzerland.

Fixed-Point Dual Gradient Projection for Embedded Model Predictive Control

Panagiotis Patrinos, Alberto Guiggiani, Alberto Bemporad

Abstract— **Although linear Model Predictive Control has gained increasing popularity for controlling dynamical systems subject to constraints, the main barrier that prevents its widespread use in embedded applications is the need to solve a Quadratic Program (QP) in real-time. This paper proposes a dual gradient projection (DGP) algorithm specifically tailored for implementation on fixed-point hardware. A detailed convergence rate analysis is presented in the presence of round-off errors due to fixed-point arithmetic. Based on these results, concrete guidelines are provided for selecting the minimum number of fractional and integer bits that guarantee convergence to a suboptimal solution within a prespecified tolerance, therefore reducing the cost and power consumption of the hardware device.**

I. Introduction

Model Predictive Control (MPC) technology is widely popular in many industrial applications due to explicit performance optimization, and its straightforward handling of constraints on inputs, outputs and states [1]. An MPC controller relies on solving a QP to minimize the difference between predicted outputs and desired set-points. The fact that a QP needs to be solved within each sampling period has limited the diffusion of MPC technologies to low-bandwidth applications where high computational resources are available, as in chemical and refinery industries. However, in the last years an increasing interest in embedded MPC solutions is spreading in many other industries, such as automotive and aerospace.

Embedding MPC on a hardware platform poses quite a few challenges, both from a system-theoretic and an optimization point of view. Specifically, the main requirements that make a QP solver suitable for embedded MPC are the following: (a) the algorithm should be simple enough to be implemented on simple hardware platforms; (b) one must be able to compute a bound on its worst-case execution time for computing a (reasonably good) solution; (c) stability and invariance guarantees for the resulting closed-loop system must be provided despite suboptimality and/or infeasibility of the solution; (d) the algorithm should be robust to low precision arithmetic, i.e., the effect of round-off errors should be small, and no overflow should occur, and one should be able to determine *a priori* the behavior of the algorithm under such hypotheses.

The authors are with IMT - Institute for Advanced Studies, Piazza San Ponziano 6, 55100 Lucca, Italy. Email: {panagiotis.patrinos, alberto.guiggiani, alberto.bemporad}@imtlucca.it.

Past embedded MPC implementations generally rely on *floating-point* representations [2], [3]. However, when trying to minimize computational effort, power consumption, and chip size, a great positive impact is given by the choice of *fixed-point* number representation [4]. Nevertheless, this significant improvement in performance comes at the price of a reduced range in which numbers can be represented and round-off errors [5]. Additional challenges arise, mainly studying error accumulation during algorithm iterations, and establishing bounds on the magnitude of the computed variables to avoid overflow.

Recently, the use of first-order methods, and in particular fast gradient methods developed by Nesterov [6], has been advocated as a viable candidate for embedded optimization-based control [7]–[10]. These methods can compute a suboptimal solution in a finite number of iterations, which can be bounded *a priori*, and they are simple enough (usually requiring only matrix-vector products) for hardware implementation. In particular, the accelerated DGP method proposed in [8], [9], called GPAD, can be applied to linear MPC problems with general polyhedral constraints and with guaranteed global *primal* convergence rates. In [11] results of [8], [9] are exploited to show how GPAD can be used in MPC to provide invariance, stability and performance guarantees in a finite number of iterations for the closed-loop system.

In this work, we propose a DGP method, which can be seen as a simplified (non-accelerated) version of GPAD, specifically tailored for fixed-point implementation, providing a convergence analysis that takes into account round-off errors, thus addressing successfully the last of the requirements described previously for embedded optimization-based control. In addition to that, we give specific guidelines on the number of decimal bits that certify the convergence to a target suboptimal solution, as well as on the number of integer bits to avoid overflow errors. The reason for limiting the analysis to the non-accelerated version of GPAD is that, in principle, accelerated methods suffer from error accumulation, as shown in [12]. In [13] the authors analyze the convergence rate of inexact gradient augmented Lagrangian methods for constrained MPC, where the source of inexactness comes from suboptimal solution of the so called inner problem. In the present work, the source of inexactness comes from round-off errors due to fixed-point implementation.

After introducing some notation in Section II, in Section III we provide the motivation of this paper,

in Section IV we summarize general theoretical results for inexact gradient projection (GP). In Section V an inexact DGP method is applied to a modified version of the dual problem and its convergence rate with respect to primal suboptimality and infeasibility is analyzed. In Section VI, the general results of the proposed inexact DGP method are applied to the case of QP based on a fixed-point implementation. Simulation results and preliminary experiments on low-cost hardware boards are presented in Section VII. Finally, conclusions are drawn in Section VIII. Due to space limitations, the proofs of the theoretical results are omitted. However, they can be made available upon request.

II. Notation

Let $\mathbb{R}$, $\mathbb{N}$, $\mathbb{R}^n$, $\mathbb{R}^{m\times n}$ denote the sets of real numbers, nonnegative integers, column real vectors of length n, and m by n real matrices, respectively. The transpose of a matrix $A \in \mathbb{R}^{m\times n}$ is denoted by A'. For any nonnegative integers $k_1 \leq k_2$ the finite set $\{k_1, \dots, k_2\}$ is denoted by $\mathbb{N}_{[k_1,k_2]}$. For $z \in \mathbb{R}^n$, $\Pi_Z(z)$ denotes its Euclidean projection on the set $Z \subseteq \mathbb{R}^n$, while $[z]_+$ denotes its Euclidean projection on the nonnegative orthant, i.e., the vector whose i-th coordinate is $\max\{z_i, 0\}$. For a vector $z \in \mathbb{R}^n$, $\|z\|$ denotes the Euclidean norm of z, while if $A \in \mathbb{R}^{m\times n}$, $\|A\|$ denotes the spectral norm of A (unless otherwise stated).

III. Motivation

When performing computations on low-cost, low-power embedded devices, the adoption of a fixed-point number representation can have a great positive impact in terms of computational speed. However, this comes at the price of a reduced precision and a reduced range when compared to floating-point representation, leading to the occurrence of round-off and overflow errors.

Suppose that an algorithm is running on a fixed-point hardware with a scaling factor 2^{-p}, where $p \in \mathbb{N}_+$ is the number of fractional bits, and assume that real numbers are represented in fixed-point by rounding to the closest value. Therefore, the resolution (i.e., the smallest representable non-zero magnitude) of a fixed-point number is equal to $2^{-(p+1)}$.

It is obvious that addition and subtraction do not result in any loss of accuracy due to rounding. However, multiplication can suffer from rounding. In specific, multiplying two fixed-point numbers $\zeta = \gamma\xi$ leads to the fixed-point representation $\mathtt{fi}(\zeta)$ of ζ, with $|\zeta - \mathtt{fi}(\zeta)| \leq 2^{-(p+1)}$.

For $x, y \in \mathbb{R}^n$ let $\mathtt{fi}(x'y) \triangleq \sum_{i=1}^n \mathtt{fi}(x_i y_i)$. Then the round-off error for the inner product of x and y can be bounded as follows

$$|x'y - \mathtt{fi}(x'y)| \leq 2^{-(p+1)}n. \tag{1}$$

If A is an $m \times n$ matrix and x is an n-vector, then

$$\|Ax - \mathtt{fi}(Ax)\|_\infty \leq 2^{-(p+1)}n. \tag{2}$$

QP algorithms based on the GP method require, at each iteration, the computation of the gradient for the cost function. In a fixed-point architecture, instead of the exact gradient $\nabla\Phi(\cdot)$, we have access to an approximate representation $\tilde{\nabla}\Phi(\cdot)$.

Convergence proofs have therefore to be reformulated in order to take into account this approximation. In addition to that, it is of interest to find direct links between the fixed-point precision and bounds on the gradient error, as well as solution quality. Finally, bounds on the magnitude of all the variables are required such that the number of integer bits can be choosn to avoid the occurrence of overflow errors. All this topics will be covered in the next sections.

IV. Inexact Gradient Projection

In this section we summarize results regarding convergence of the classical GP method in the case where at every iteration, instead of the gradient, only an *inexact oracle* providing an *approximate* first-order information of the cost function is available . The details of the following analysis can be found in [12]. Consider the problem

$$\Phi^\star = \min_{y\in Y} \Phi(y), \tag{3}$$

where Y is a closed convex set, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ is convex, L_Φ-smooth, i.e., there exists a $L_\Phi > 0$ such that

$$\|\nabla\Phi(y) - \nabla\Phi(w)\| \leq L_\Phi\|y - w\|, \quad y, w \in \mathbb{R}^m.$$

We assume that $Y^\star \triangleq \operatorname{argmin}_{y\in Y} \Phi(y)$ is nonempty. The goal is to find an approximate solution of (3) by applying the GP method. However, it is assumed that the gradient of Φ cannot be computed exactly. Instead, we have at our disposal an inexact oracle as defined in [12].

We call the first-order $(0, L_\Phi)$-oracle, $(\Phi(y), \nabla\Phi(y))$, the *exact oracle*. Now, there is a difference in the implementation of GP with the exact and inexact oracle. The one with the exact oracle is the standard GP algorithm

$$y_{(\nu+1)} = \Pi_Y(y_{(\nu)} - \tfrac{1}{L_\Phi}\nabla\Phi(y_{(\nu)})), \tag{4}$$

whereas GP with an inexact oracle is

$$y_{(\nu+1)} = \Pi_Y(y_{(\nu)} - \tfrac{1}{L}s_{\delta,L}(y_{(\nu)})). \tag{5}$$

The next theorem provides convergence rate estimates for the inexact GP scheme (5) [12, Theor. 4].

Theorem 1: Let $\{y_{(\nu)}\}_{\nu\in\mathbb{N}}$ be generated by iterating (5) from any $y_{(0)} \in Y$ and let $\bar{y}_{(\nu+1)} \triangleq \frac{1}{\nu+1}\sum_{i=0}^{\nu} y_{(i+1)}$. Then

$$\Phi(\bar{y}_{(\nu+1)}) - \Phi(y^\star) \leq \tfrac{L}{2(\nu+1)}\|y^\star - y_{(\nu)}\|^2 + \delta. \tag{6}$$

V. Inexact Dual Gradient Projection

In this section we analyze the inexact DGP method when applied to solve

$$V^\star = \min_{z\in\mathbb{R}^n}\{V(z) | g(z) \leq 0\}, \tag{7}$$

which we call the *primal problem*. In (7), $V : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable, strongly convex function with convexity parameter κ_V, i.e.,

$$(\nabla V(z_1) - \nabla V(z_2))'(z_1 - z_2) \geq \kappa_V\|z_1 - z_2\|^2,$$

for all $z_1, z_2 \in \mathbb{R}^n$, and $g(z) = Az - b$, $A \in \mathbb{R}^{m\times n}$. The unique solution of (7) is denoted by $z^\star$. Suppose now

that the main goal is to compute an $(\varepsilon_V, \varepsilon_g)$-optimal solution for (7), defined as follows.

Definition 1: Consider two nonnegative constants ε_V, ε_g. Vector z is an $(\varepsilon_V, \varepsilon_g)$-optimal solution for (7) if

$$V(z) - V^\star \leq \varepsilon_V, \tag{8a}$$

$$\|[g(z)]_+\|_\infty \leq \varepsilon_g. \tag{8b}$$

The (negative of the) *dual problem* of (7) can be expressed as (3), with

$$\Phi(y) = -\min_{z\in\mathbb{R}^n} \mathcal{L}(z,y) \tag{9}$$

being the (negative) dual function, where

$$\mathcal{L}(z,y) = V(z) + y'g(z),$$

is the Lagrangian of (7) and $Y \triangleq \mathbb{R}^m_+$ is the dual feasible set. For the rest of the paper we assume that there is no duality gap, i.e., $V^\star = -\Phi^\star$. This assumption is fulfilled if, for example, Problem (7) is a convex quadratic program. Since V is strongly convex, $z_y^\star = \operatorname{argmin}_{z\in\mathbb{R}^n} \mathcal{L}(z,y)$ is unique, and $\Phi : \mathbb{R}^m \to \mathbb{R}$ is convex, L_Φ-smooth with its gradient given by

$$\nabla\Phi(y) = -g(z_y^\star). \tag{10}$$

Therefore, the GP algorithm applied to the dual problem (9) becomes (for a given $y_{(0)}$)

$$z_{(\nu)} = \operatorname{argmin}_{z\in\mathbb{R}^n} \mathcal{L}(z, y_{(\nu)}) \tag{11a}$$

$$y_{(\nu+1)} = \left[y_\nu + \tfrac{1}{L_\Phi} g(z_{(\nu)})\right]_+ \tag{11b}$$

Next, assume that for every $y \in Y$, instead of $\nabla\Phi(y) = -g(z_y^\star)$, one can only calculate an approximate gradient

$$\tilde{\nabla}\Phi(y) = -g(z_y) + \xi, \tag{12}$$

where

$$\|z_y - z_y^\star\| \leq \epsilon_z, \quad \|\xi\| \leq \epsilon_\xi, \tag{13}$$

for given positive constants ϵ_z, ϵ_ξ.

A. Modified Primal-Dual Pair

The goal is to construct a first-oder inexact oracle for Φ (cf. (9)) with $s_{\delta,L}(y) = \tilde{\nabla}\Phi(y)$. Convergence rate results for GP methods in the presence of an additive disturbance ξ require the constraint set Y of (3) to be bounded [14], [15]. For this reason, the dual problem (3) will be modified in order to have a bounded constraint set. Of course, the new dual problem will correspond to a modified primal problem, different from (7), therefore one should carefully draw conclusions about the connection of the solution sets of the original and of the modified primal-dual pair. We now proceed in describing this construction in detail.

Let $d \in \mathbb{R}^m$ be such that its i-th element satisfies

$$d_i \geq \max\{y_i^\star, 1\} \tag{14}$$

for some $y^\star \in Y^\star$, and

$$Y_\alpha \triangleq \{y \in \mathbb{R}^m |\ 0 \leq y \leq \alpha d\}, \quad \alpha > 1. \tag{15}$$

Furthermore, let $D \triangleq \|d\|$ and $D_\alpha \triangleq \max_{y_1,y_2\in Y_\alpha} \|y_1 - y_2\| = \alpha D$, the diameter of Y_α. Obviously $Y_\alpha \subseteq Y$ and $Y_\alpha \cap Y^\star \neq \emptyset$. Since there is no duality gap, by the Lagrangian Saddle Point Theorem (see e.g. [16, Prop. 6.2.4]) the set of primal-dual solutions are exactly the saddle points of the Lagrangian on $\mathbb{R}^n \times \mathbb{R}^m_+$, i.e., $z^\star = \operatorname{argmin}\{V(z) | g(z) \leq 0\}$, $y^\star \in Y^\star$ if and only if they satisfy the saddle point inequality

$$\mathcal{L}(z^\star, y) \leq \mathcal{L}(z^\star, y^\star) \leq \mathcal{L}(z, y^\star),\ \forall\ z \in \mathbb{R}^n,\ y \in Y.$$

By definition of Y_α and the saddle point inequality, the set of saddle points of $\mathcal{L}$ on $\mathbb{R}^n \times Y_\alpha$ is nonempty and equal to $\{z^\star\} \times Y_\alpha^\star$, where $Y_\alpha^\star \triangleq Y^\star \cap Y_\alpha$. By [16, Prop. 2.6.1], this set of saddle points is completely characterized by

$$\begin{aligned} z^\star &= \operatorname*{argmin}_{z\in\mathbb{R}^n} \max_{y\in Y_\alpha} \mathcal{L}(z,y) \\ &= \operatorname*{argmin}_{z\in\mathbb{R}^n} V(z) + \alpha\textstyle\sum_{i=0}^m d_i [g_i(z)]_+, \end{aligned} \tag{16}$$

and

$$Y_\alpha^\star = \operatorname*{argmax}_{y\in Y_\alpha} \min_{z\in\mathbb{R}^n} \mathcal{L}(z,y) = \operatorname*{argmin}_{y\in Y_\alpha} \Phi(y). \tag{17}$$

Interestingly, from (16) it is apparent that $V(z) + \alpha\sum_{i=0}^m d_i [g_i(z)]_+$ furnishes a nondifferentiable *exact penalty function* for (7), a fact that is well known, see e.g. [17, Sec. 4.3.1]. Furthermore, the new dual problem (17) has a bounded constraint set.

Remark 1: In principle, determining a vector d such that (14) holds requires one to know bounds on the elements of a dual optimal solution. If (7) is a parametric QP, as in embedded linear MPC, then tight uniform bounds (valid for every admissible parameter vector) can be computed using techniques described in [18]. In fact, one has to compute such bounds anyway, since they are imperative for determining the worst-case number of iterations, and consequently the worst-case running time of the algorithm, a central concern in embedded optimization (see e.g. [7]–[10]).

B. Inexact Oracle

We are now ready to derive an inexact oracle for Φ on Y_α under assumptions (12), (13).

Proposition 1: Consider Φ given by (9), with $g(z) = Az - b$. Then, the pair

$$\Phi_{\delta,L}(y) = -\mathcal{L}(z_y, y) - \alpha D \epsilon_\xi, \tag{18a}$$

$$s_{\delta,L}(y) = \tilde{\nabla}\Phi(y) = -g(z_y) + \xi \tag{18b}$$

furnishes a (δ_α, L)-oracle for Φ on Y_α, where $\delta_\alpha \triangleq L_V \epsilon_z^2 + 2\alpha D \epsilon_\xi$, $L \triangleq \frac{2}{\kappa_V}\|A\|^2$.

Notice that the oracle error δ_α decreases with α, achieving its minimum value for $\alpha = 1$. Furthermore, the bounding of the dual feasible set is essential (cf. (15)), otherwise it would not be possible to bound quantities such as $\|w - y\|$, for any $w, y \geq 0$.

C. Primal Convergence Rates

Under the assumptions imposed by (12), (13), the ν-th iteration of the inexact DGP scheme applied to Problem (17) with the first-order oracle given by Proposition 1 is

$$y_{(\nu+1)} = \Pi_{Y_\alpha}(y_{(\nu)} + \tfrac{1}{L}(g(z_{(\nu)}) + \xi_{(\nu)})), \tag{19}$$

with $z_{(\nu)}, \xi_{(\nu)}$ s.t. $\|z_{(\nu)} - z^\star_{y_{(\nu)}}\| \leq \epsilon_z, \|\xi_{(\nu)}\| \leq \epsilon_\xi$.

The Euclidean projection onto Y_α is very easy to compute, since for $w \in \mathbb{R}^m$ we have $\Pi_{Y_\alpha}(w) = \max\{\min\{w, \alpha d\}, 0\}$. Notice that Theorem 1, [12, Theor. 4] readily gives dual suboptimality bounds for inexact DGP method (19). However, in MPC applications one is typically interested in primal suboptimality and infeasibility bounds.

Therefore, we will next derive global convergence rates to primal optimality and primal feasibility for the averaged primal sequence

$$\bar{z}_{(\nu)} \triangleq \frac{1}{\nu+1}\sum_{i=0}^{\nu} z_{(i)}. \tag{20}$$

Theorem 2: Let $\{y_{(\nu)}, z_{(\nu)}\}$ be generated by iterating (19), starting from any $y_{(0)} \in Y_\alpha$. If $\alpha>1$, then for any $\nu \in \mathbb{N}$

$$\|[g_i(\bar{z}_{(\nu)})]_+\|_\infty \leq \frac{\alpha^2}{\alpha-1}\frac{LD^2}{2(\nu+1)} + \delta_\alpha^g, \tag{21}$$

where $\delta_\alpha^g \triangleq \frac{1}{\alpha-1}L_V\epsilon_z^2 + \frac{\alpha}{\alpha-1}2D\epsilon_\xi$.

Notice that there is a trade-off between the constant of the $O(1/\nu)$ term determining the convergence rate to feasibility, and the maximum level of infeasibility that one is able to tolerate, asymptotically. As $\alpha \to \infty$, δ_α^g approaches its infimum, $2D\epsilon_\xi$, while $\frac{\alpha^2}{\alpha-1} \to \infty$. By choosing $\alpha = 2$ (the one that minimizes $\frac{\alpha^2}{\alpha-1}$) we arrive at

$$\|[g(\bar{z}_{(\nu)})]_+\|_\infty \leq \frac{2LD^2}{\nu+1} + L_V\epsilon_z^2 + 4D\epsilon_\xi. \tag{22}$$

Theorem 3: Let $\{y_{(\nu)}, z_{(\nu)}\}$ be generated by iterating (19), starting from any $y_{(0)} \in Y_\alpha$. Then

$$V(\bar{z}_{(\nu)}) - V^\star \leq \frac{L}{2(\nu+1)}(\|y^\star\|^2 + \|y_{(0)}\|^2) + \delta_\alpha, \tag{23a}$$

$$V(\bar{z}_{(\nu)}) - V^\star \geq -\left(\frac{\alpha^2}{\alpha-1}\frac{LD^2}{2(\nu+1)} + \delta_\alpha^g\right)D. \tag{23b}$$

Notice that the constant of the $O(1/\nu)$ term in (23a) is independent of α. In fact, if iterations (19) start from $y_{(0)} = 0$, then the cost $V(\bar{z}_{(\nu)})$ is always lower than $V^\star + \delta_\alpha$, the best achievable by the corresponding scheme asymptotically, as it is shown below. In that case, one has to worry only about feasibility.

Corollary 1: Let $\{y_{(\nu)}, z_{(\nu)}\}$ be generated by iterating (19) from any $y_{(0)} \in Y_\alpha$. Then

$$V(\bar{z}_{(\nu)}) - V^\star \leq \delta_\alpha, \quad \forall \nu \in \mathbb{N}. \tag{24}$$

We will next derive the value of the user-defined parameter α that achieves the fastest convergence rate to an $(\varepsilon_V, \varepsilon_g)$-solution, given oracle parameters ϵ_z, ϵ_ξ. For simplicity, we assume that the initial iterate is equal to the zero vector, i.e., $y_{(0)} = 0$. In that case one should only worry about convergence to primal feasibility since, due to Corollary 1, $V(\bar{z}_{(\nu)}) - V^\star \leq \delta_\alpha$, for every $\nu \in \mathbb{N}_+$.

Theorem 4: Suppose that $\varepsilon_g > 2D\epsilon_\xi$ and $\varepsilon_V > \frac{\varepsilon_g(L_V\epsilon_z^2 + 2D\epsilon_\xi)}{\varepsilon_g - 2D\epsilon_\xi}$. Then an $(\varepsilon_V, \varepsilon_g)$-solution is obtained by iterating (19) from $y_{(0)} = 0$ with $\alpha = \alpha^\star$,

$$\alpha^\star \triangleq \min\left\{\frac{2(\varepsilon_g + L_V\epsilon_z^2)}{\varepsilon_g - 2D\epsilon_\xi}, \frac{\varepsilon_V - L_V\epsilon_z^2}{2D\epsilon_\xi}\right\}, \tag{25}$$

no more than $\nu(\alpha^\star)$ times, where

$$\nu(\alpha) = \frac{LD^2\alpha^2}{2(\varepsilon_g - 2D\epsilon_\xi)\alpha - 2(\varepsilon_g + L_V\epsilon_z^2)} - 1. \tag{26}$$

For the nominal case ($\epsilon_z = \epsilon_\xi = 0$), from (25) we obtain $\alpha^\star = 2$.

D. Maximum Admissible Oracle Errors ϵ_z, ϵ_ξ

Based on the results of Section V, we will give explicit formulas for the maximum admissible oracle errors ϵ_z, ϵ_ξ as a function of solution accuracy $\varepsilon_V, \varepsilon_g$, and consequently for the maximum number of iterations that need to be executed.

The question just posed is of significant importance in embedded optimization-based control, like MPC, for the following reason. Given hard real-time constraints dictated by hardware specifications and sampling time, as well as sufficiently small values for $\varepsilon_V, \varepsilon_g$ that guarantee closed-loop stability (see [11]), one wants to determine the smallest allowable oracle precision (maximum allowable values for ϵ_z, ϵ_ξ) that achieves the aforementioned requirements. As it will become clear in Section VI, ϵ_z, ϵ_ξ correspond to round-off errors due to fixed-point arithmetic. The smaller the number of fractional bits is (i.e., the larger the maximum allowable oracle errors), the smaller the execution time and the power consumption of the hardware device will be.

Again, for simplicity we assume that $y_{(0)} = 0$. Moreover, we suppose that $\epsilon_\xi = \beta\epsilon_z$, for some $\beta > 0$. This last assumption is justified in Section VI. Finally, it is assumed that $\alpha = 2$, since for small ϵ_z, ϵ_ξ this is usually the best choice. First, from (22), we have

$$\frac{2LD^2}{\nu+1} + L_V\epsilon_z^2 + 4D\beta\epsilon_z \leq \varepsilon_g. \tag{27}$$

By solving with respect to ϵ_z, we arrive at

$$\epsilon_z \leq \sqrt{\frac{\varepsilon_g}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2 - \frac{2LD^2}{L_V(\nu+1)}} - \frac{2D\beta}{L_V}. \tag{28}$$

Due to Corollary 1, one must have $\delta_2 \leq \varepsilon_V$, or

$$\epsilon_z \leq \sqrt{\frac{\varepsilon_V}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2} - \frac{2D\beta}{L_V}. \tag{29}$$

Letting $\nu \to \infty$ in (28), and taking into account that $\sqrt{\varepsilon/L_V + (2D\beta/L_V)^2}$ is increasing as a function of ε, we conclude that, in order to be able to converge asymptotically to an $(\varepsilon_V, \varepsilon_g)$-solution, ϵ_z must satisfy

$$\epsilon_z < \sqrt{\frac{\varepsilon}{L_V} + \left(\frac{2D\beta}{L_V}\right)^2} - \frac{2D\beta}{L_V}, \tag{30}$$

where $\varepsilon = \min\{\varepsilon_g, \varepsilon_V\}$. It is worth mentioning that the maximum oracle error ϵ_z, that allows one to reach accuracy ε decreases as $O(\sqrt{\varepsilon})$, slower than $O(\varepsilon)$ for $\varepsilon < 1$ (which is usually the case of interest).

VI. Fixed-Point DGP for QPs

The theory presented in Section V allows us to analyze the fixed-point implementation of the DGP algorithm defined by (11) for strictly convex quadratic programs. Consider problem (7) when $V(z) = \frac{1}{2}z'Qz + q'z$, $g(z) = Az - b$, with $\kappa_V = \lambda_{\min}(Q) > 0$ and $A \in \mathbb{R}^{m\times n}$. The dual problem (modulo a sign change) is (3) with $\Phi(y) = \frac{1}{2}y'Hy + h'y$, and $Y = \mathbb{R}_+^m$, where $H = AQ^{-1}A'$, $h = AQ^{-1}q + b$. Furthermore,

$$z_y^\star = Ey + e, \tag{31}$$

where $E = -Q^{-1}A'$, $e = -Q^{-1}q$.

Therefore, the DGP iterations (11) lead to the following algorithm ($y_{(0)} = 0$)

$$z_{(\nu)} = Ey_{(\nu)} + e,\ g_{(\nu)} = Az_{(\nu)} - b \tag{32a}$$

$$y_{(\nu+1)} = [y_{(\nu)} + \tfrac{1}{L}g_{(\nu)}]_+ \tag{32b}$$

with stopping criterion

$$\|[A\bar{z}_{(\nu)} - b]_+\|_\infty \leq \epsilon_g, \tag{32c}$$

$\bar{z}_{(\nu)}$ given by (20) and $y_{(0)} = 0$.

A. Fixed-point implementation

Let the algorithm defined by (32) be embedded on a fixed-point hardware with a scaling factor 2^{-p}, where $p \in \mathbb{N}_+$ is the number of fractional bits.

In a fixed-point architecture, for a given $y \in \mathbb{R}^m$, instead of the gradient $\nabla\Phi(y) = -g(z_y^\star)$, where $z_y^\star$ is given by (31), we have access to $\tilde{\nabla}\Phi(y)$ of the form (12), with $z_y = \texttt{fi}(Ey + e)$, and $\xi = g(z_y) - \texttt{fi}(Az_y - b)$. Due to (2), the vectors ξ, z_y satisfy (13) with

$$\epsilon_z = 2^{-(p+1)} m\sqrt{n}, \tag{33}$$

$$\epsilon_\xi = 2^{-(p+1)} n\sqrt{m}. \tag{34}$$

Since $L = \frac{2}{\kappa_V}\|A\|^2$, by properly scaling the problem matrices we can assume that $L = 1$, therefore there is no round-off error in computing the product $\frac{1}{L}g_{(\nu)}$.

The ν-th iteration of the inexact DGP scheme (19) implemented on the fixed-point hardware platform is

$$z_{(\nu)} = \texttt{fi}(Ey_{(\nu)} + e),\ g_{(\nu)} = \texttt{fi}(Az_{(\nu)} - b), \tag{35a}$$

$$y_{(\nu+1)} = \max\{\min\{y_{(\nu)} + \tfrac{1}{L}g_{(\nu)}, \alpha d\}, 0\}. \tag{35b}$$

B. Number of fractional bits

We now provide explicit bounds on the number of fractional bits required to grant convergence of (35) to a target primal suboptimal solution satisfying (32c).

Corollary 2: Let $\{z_{(\nu)}\}$ be generated by iterating (35), with $y_{(0)} = 0$. Assume that real numbers are rounded to the closest fixed-point value. Then, the algorithm converges asymptotically to an $(\varepsilon_g, \varepsilon_V)$-solution, with $\varepsilon_g > 2D\epsilon_\xi$ and $\varepsilon_V > \frac{\varepsilon_g(L_V\epsilon_z^2 + 2D\epsilon_\xi)}{\varepsilon_g - 2D\epsilon_\xi}$, if the number of fractional bits p is such that

$$p \geq \log_2 \frac{m\sqrt{n}}{\sqrt{\frac{\varepsilon}{L_V} + \frac{n}{m}\left(\frac{2D}{L_V}\right)^2} - \sqrt{\frac{n}{m}}\frac{2D}{L_V}} - 1, \tag{36}$$

where $\varepsilon = \min\{\varepsilon_g, \varepsilon_V\}$.

C. Number of integer bits

Together with round-off errors, another key issue that arises while embedding computations on fixed-point architectures is the occurrence of overflow errors, given by the limited range for number representation. In particular, if the number of bits for the integer part equals r, the computed variables can only assume values in $\left[-2^{r-1}, 2^{r-1} - 1\right]$, where the asymmetry is given by the presence of the zero element. The following corollary sets precise guidelines for choosing a number of integer bits that is sufficiently large to avoid overflows.

Corollary 3: Let the iterations in (35) be run on a fixed-point architecture with r bits for the integer part and $y_{(0)} = 0$. Then, occurence of overflow errors is avoided if r is chosen such that

$$r \geq \log_2\left(\max\{\hat{y}, \hat{z}, \hat{g}\} + 1\right) + 1, \tag{37}$$

where $\hat{y} = \alpha\|d\|_\infty$, $\hat{z} = \|E\|_\infty\hat{y} + \|e\|_\infty$, $\hat{g} = \|A\|_\infty\hat{z} + \|b\|_\infty$.

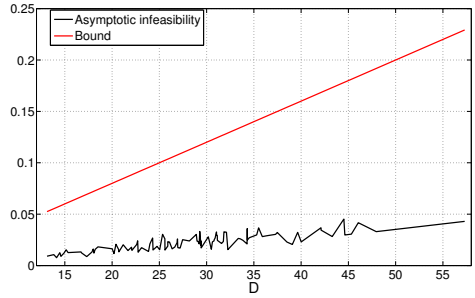

Fig. 1. Asymptotic infeasibility values compared to theoretical bound (21)

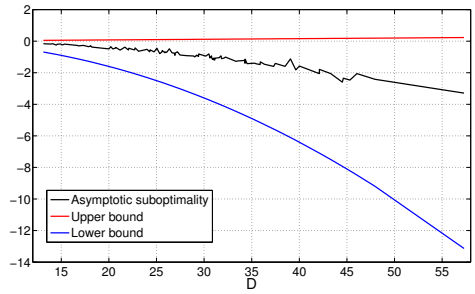

Fig. 2. Asymptotic suboptimality values compared to theoretical bounds (23a) and (23b)

VII. Simulations

A. Bounds on primal infeasibility and suboptimality

The purpose of the first simulation is to test the tightness for the primal infeasibility and suboptimality bounds given by Theorems 2 and 3, respectively. The analysis was performed on a worst-case scenario, running iterations (19) with $\|\xi_{(\nu)}\| = \epsilon_\xi$ to solve 100 randomly generated QP problems, with 10 optimization variables and 20 constraints. The goal was to compare error bound terms δ_α and δ_α^g with the practical asymptotic values of the primal infeasibility (Figure 1) and suboptimality (Figure 2) for $\nu \to \infty$. Different trials are ordered for increasing values of D, term proportional to the constraint set diameter. Simulation results show an acceptable tightness for bounds, as they exceed the practical values by a factor between 3.33 and 8.32 for infeasibility, and between 3.05 and 5.74 for suboptimality. In addition, the linear (for infeasibility) and quadratic (for suboptimality) theoretical dependencies on D are reflected in the experimental results.

B. Bounds on iteration count

The following simulation is performed to test the tightness of the theoretical bound on the number of iterations given by (26). We let Algorithm (35) run on Matlab R2012b and Fixed-Point Toolbox v.3.6 on a Mid-2012 Macbook Pro Retina running OSX 10.8.2 to solve various random QPs (sizes are equal to 4 and 8 for the primal and the dual, respectively). In Figure 3, the practical number of iterations needed to reach decreasing target infeasibilities is plotted against

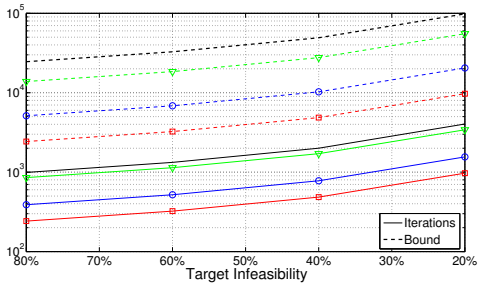

Fig. 3. Comparison between the actual number of iterations and the theoretical bounds predicted by (26) calculated on random QP problems.

TABLE I

FIXED-POINT HARDWARE IMPLEMENTATION

Size	T $[ms]$	TPI $[\mu s]$	Size $[KB]$
10/20	22.9	226	15
20/40	52.9	867	17
40/80	544.9	3382	27
60/120	1519.8	7561	43

the theoretical bound; different colors and markers are chosen for different QP solutions.

C. Hardware Implementation

Finally, Algorithm (35) has been implemented on a 32-bit Atmel SAM3X8E ARM Cortex-M3 processing unit; this chipset operates at a maximum speed of 84 MHz and comes with 512 KB of flash memory and 100 KB of RAM. The microcontroller was assigned to solve random QP problems of increasing size, ranging from 10 to 60 primal variables and 20 to 120 primal constraints. The algorithm was stopped upon reaching a suboptimal solution bounded by 10% primal infeasibility. Table I shows the results when a fixed-point number representation is adopted, with 8 bits for the decimal part and 7 bits for the integer part. For each problem size we report convergence time, average time per iteration (TPI) and size of the binary code; the latter plays an important role in embedded applications, where usually a limited amount of memory is available. The same simulation are repeated switching to floating-point; results in Table II show up to 4 times longer computations and up to 2 times larger code size, with gains increasing as the problem becomes larger.

VIII. CONCLUSIONS AND FUTURE WORK

This paper has proposed a DGP method for embedded MPC on hardware with fixed-point arithmetic. Concrete and theoretically-proven guidelines for selecting the minimum number of decimal and integer bits that guarantee favorable convergence properties are provided. Future work includes quantifying the effect of fixed-point arithmetic on accelerated versions of the the DGP method and perhaps modifications of it to achieve the optimal trade-off between convergence rate and round-off error accumulation, as well as experimental implementation and testing of the algorithm on real hardware platforms.

TABLE II

FLOATING-POINT HARDWARE IMPLEMENTATION

Size	T $[ms]$	TPI $[\mu s]$	Size $[KB]$
10/20	88.6	974	16
20/40	220.1	3608	21
40/80	2240	13099	40
60/120	5816	30450	73

REFERENCES

[1] A. Bemporad, "Model predictive control design: New trends and tools," in *Decision and Control, 2006 45th IEEE Conference on*, 2006, pp. 6678–6683.

[2] K. V. Ling, S. P. Yue, and J. Maciejowski, "A FPGA implementation of model predictive control," in *American Control Conference*, 2006, pp. 1930–1935.

[3] G. Knagge, A. Wills, A. Mills, and B. Ninness, "ASIC and FPGA implementation strategies for model predictive control," in *Proc. European Control Conference*, 2009.

[4] E. Kerrigan, J. Jerez, S. Longo, and G. Constantinides, "Number representation in predictive control," in *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, M. Lazar and F. Allgower, Eds., August 2012, pp. 60–67.

[5] J. Wilkinson, *Rounding Errors in Algebraic Processes*. Dover Publications, 1994.

[6] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers, 2004.

[7] 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 and European Control Conf.*, Orlando, USA, 2011, pp. 5223–5229.

[8] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for linear model predictive control," in *IEEE 51st Conference on Decision and Control*, 2012, pp. 662–667.

[9] A. Bemporad and P. Patrinos, "Simple and certifiable quadratic programming algorithms for embedded linear model predictive control," in *Proc. 4th IFAC Nonlinear Model Predictive Control Conference*, M. Lazar and F. Allgower, Eds., 2012.

[10] P. Giselsson, "Execution time certification for gradient-based optimization in model predictive control," in *IEEE 51st Annual Conference on Decision and Control (CDC)*, 2012, pp. 3165–3170.

[11] M. Rubagotti, P. Patrinos, and A. Bemporad, "Stabilizing embedded MPC with computational complexity guarantees," in *Proc. 12th European Control Conference (ECC)*, 2013.

[12] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *CORE Discussion Papers,(2011/02)*, 2011.

[13] V. Nedelcu and I. Necoara, "Iteration complexity of an inexact augmented lagrangian method for constrained mpc," in *IEEE 51st Conference on Decision and Control*, 2012, pp. 650–655.

[14] A. d'Aspremont, "Smooth optimization with approximate gradient," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1171–1183, 2008.

[15] O. Devolder, "Stochastic first order methods in smooth convex optimization," *CORE Discussion Papers 2012*, vol. 9, 2012.

[16] D. Bertsekas, A. Nedic, and A. Ozdaglar, *Convex analysis and optimization*. Athena Scientific, 2003.

[17] D. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[18] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for linear model predictive control," provisionally accepted in IEEE Transactions on Automatic Control, 2013.