Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Fixed-Point Implementation of a Proximal Newton Method for Embedded Model Predictive Control

Alberto Guiggiani * **Panagiotis Patrinos** * **Alberto Bemporad** *

* *IMT - Institute for Advanced Studies, Piazza San Ponziano 6, 55100 Lucca, Italy (e-mail: {alberto.guiggiani, panagiotis.patrinos, alberto.bemporad}@imtlucca.it).*

Abstract: Extending the success of model predictive control (MPC) technologies in embedded applications heavily depends on the capability of improving quadratic programming (QP) solvers. Improvements can be done in two directions: better algorithms that reduce the number of arithmetic operations required to compute a solution, and more efficient architectures in terms of speed, power consumption, memory occupancy and cost. This paper proposes an implementation with fixed-point arithmetic of a proximal Newton method to solve optimization problems arising in input-constrained MPC. The main advantages of the algorithm are its fast asymptotic convergence rate and its relatively low computational cost per iteration since it the solution of a small linear system is required.
A detailed analysis on the effects of quantization errors is presented, showing the robustness of the algorithm with respect to finite-precision computations. A hardware implementation with specific optimizations to minimize computation times and memory footprint is also described, demonstrating the viability of low-cost, low-power controllers for high-bandwidth MPC applications. The algorithm is shown to be very effective for embedded MPC applications through a number of simulation experiments.

Keywords: Predictive control, Convex optimization, Embedded systems, Model-based control, Implementation, Gradient methods, Quantization errors, Overflow

1. INTRODUCTION

Model Predictive Control (MPC) is a mature and well-estabilished technology in many application areas (Bemporad (2006); Qin and Badgwell (2003)). It supports multi-input, multi-output process models, grants a direct performance optimization, and inherently handles constraints on inputs, outputs and states (Rawlings and Mayne (2009)).

In standard MPC approaches, the control action is generated from the optimal solution of a Quadratic Programming (QP) problem where a performance index is minimized, usually a combination of the controller effort and the deviation of predicted outputs from reference trajectories. This optimization problem is parametrized with respect to the current state measurement and needs to be solved within each sampling period. This fact has constrained the applicability of MPC strategies to low-bandwidth processes, as in chemical industries and refineries. To overcome this problem, a possibility is to pre-compute an explicit, piecewise-affine mapping between the state space and the optimal inputs (Bemporad et al. (2002); Patrinos and Sarimveis (2010)). Approximate explicit MPC methods tailored to FPGAs (Field Programmable Gate Arrays) were proposed by Bemporad et al. (2011) based on simplicial partitions. However, as the QP size increases, this approach becomes infeasible, mainly due to memory requirements.

In the last years there has been an increasing interest and effort to extend predictive control methods to a broader range of applications, including high-bandwidth systems with low-power embedded controllers (see Jerez et al. (2011); Knagge et al. (2009); Ling et al. (2008); Vouzis et al. (2009); Wills et al. (2012)). Those implementations rely on *floating-point* number representations, where real-world values are coded into binary words with variable scaling.

Within embedded setups, using *fixed-point* arithmetic grants an optimal utilization of the computing and power resources (Kerrigan et al. (2012)). However, this comes at the price of numerical issues with a critical impact on the stability and the convergence of the optimization algorithm. More precisely, the number representation space: (1) is quantized, causing round-off errors to occur when performing arithmetical operations (except for additions and subtractions), and (2) has a lower and an upper bound, leading to underflow and overflow errors. Although those issues arise for floating-point arithmetic too, their effect is amplified in the fixed-point case. Therefore, implementing an optimization solver on embedded hardware with fixed-point arithmetic requires a QP algorithm that is robust to finite precision computations. Moreover, it becomes important to study the propagation of errors between algorithm iterations, as well as to define design guidelines that guarantee the avoidance of overflow and underflow errors.

Very recently, successful applications of QP solvers with fixed-point arithmetic to embedded MPC were reported. Jerez et al. (2013) proposed an FPGA implementation of the Fast Gradient method (cf. Nesterov (2004)), Longo et al. (2012) an interior-point algorithm, and Patrinos et al. (2013) a dual gradient projection method.

In this paper, we present an implementation with fixed-point arithmetic of a QP solver based on the proximal Newton method of Patrinos and Bemporad (2013). We focus our efforts in analyzing crucial issues resulting from

low-precision computations, and propose optimizations to enhance algorithm efficiency and robustness. Moreover, we perform an in-depth comparison against gradient-based approaches, both in terms of computational complexity and solution accuracy. We assess algorithm performance when employed as a solver in an MPC controller, and finally propose a hardware implementation on a low-power, low-cost computing device. The applicability of the algorithm to embedded MPC applications is confirmed by a number of numerical experiments.

This paper is organized as follows. After a brief description of the adopted notation in Section 2, Section 3 introduces the problem setting and the motivation behind this work. Section 4 details the proximal Newton procedure, explains the basic concepts of fixed-point arithmetic, and investigates the impact of low-precision computation on the algorithm execution. In Section 5 two algorithm optimizations are discussed, while Section 6 shows the results of simulations exploring computational complexity, solution accuracy, and an aircraft control application. A hardware implementation is proposed in Section 7. Finally, conclusions are drawn in Section 8.

2. 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 a vector $z \in \mathbb{R}^n$, z_i denotes its *i-th* component, $\|z\|$ the Euclidean norm, and $[z]_{\underline{z}}^{\overline{z}}$ the Euclidean projection on the box $[\underline{z}, \overline{z}]$, i.e. $[z]_{\underline{z}}^{\overline{z}} = \min\{\max\{z, \underline{z}\}, \overline{z}\}\}$, where $\underline{z}, \overline{z} \in \mathbb{R}^n$ are lower and upper bounds for z, respectively. If $A \in \mathbb{R}^{m\times n}$, $\|A\|$ denotes the spectral norm of A (unless otherwise stated), A_i the *i-th* row, and A_{ij} the element at row i and column j. For a symmetric positive definite matrix Q, $\lambda_{\min}(Q)$, $\lambda_{\max}(Q)$ denote its smallest and largest eigenvalues, respectively.

3. PROBLEM SETUP AND MOTIVATION

Consider the following input-constrained, discrete-time LTI system

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \\ \underline{u} &\le u(t) \le \overline{u},\ \forall t \in \mathbb{N}, \end{aligned} \tag{1}$$

where $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $u(t) \in \mathbb{R}^{n_u}$ the input vector, and A, B are real-valued matrices of appropriate dimensions.

From (1) it is possible to formulate a receding-horizon optimal control problem over N prediction steps (cf. Maciejowski (2002)) in the following condensed form

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2} z'Qz + (F_x x)'z \\ \text{subject to} \quad & z \in \mathcal{Z}, \end{aligned} \tag{2}$$

where x is the measured system state, $z \in \mathbb{R}^n$ denotes the sequence of inputs over the control horizon ($n = Nn_u$), $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive definite (SPD) matrix, $F_x \in \mathbb{R}^{n\times n_x}$ and $\mathcal{Z} = [\underline{u}, \overline{u}] \times ... \times [\underline{u}, \overline{u}]$ is the feasible set.

The goal is to implement an algorithm to solve (2) each time a new measurement of the state x of the controlled system arrives. The solution should be computed sufficiently fast to be suitable for high-bandwidth applications (e.g., automotive and aerospace, where sampling periods are usually in the range $10 - 100$ ms), even in control frameworks where computing capabilities and power consumptions may be constrained.

4. FIXED-POINT PROXIMAL NEWTON METHOD

In this section we summarize the proximal Newton method of Patrinos and Bemporad (2013) for the solution of the bound-constrained QP (2). Then, we analyze quantization and overflow errors occurring with fixed-point number representation.

4.1 Proximal Newton Algorithm

A computationally efficient proximal Newton method for convex, possibly non-smooth, composite optimization problems has been recently introduced by Patrinos and Bemporad (2013). The proposed approach is attractive for embedded applications since it retains the low number of iterations typical of Newton-based methods, and concurrently lowers per-iteration complexity requiring the solution of a linear system on a reduced-order problem. Algorithm 1 is the proximal Newton method applied to solve the quadratic programming problem (2). It is based on the idea that problem (2) is equivalent to minimizing the real-valued, continuously differentiable convex function

$$\begin{aligned} F_\gamma(z) = & V(z) - \\ & -\tfrac{\gamma}{2}\|\nabla V(z)\|^2 + \tfrac{1}{2\gamma}\|z - \gamma\nabla V(z) - [z - \gamma\nabla V(z)]_{\underline{z}}^{\overline{z}}\|^2, \end{aligned}$$

where V denotes the cost function of problem (2) (we omit the dependence on x for clarity), provided that the parameter γ is smaller than $1/\lambda_{\max}(Q)$. The procedure requires as inputs the Hessian matrix Q, the linear term $q = F_x x$, and the box constraints $(\underline{z}, \overline{z})$ on the optimization vector. Additional tuning parameters are the value of γ and the line search parameter $\sigma \in (0, \frac{1}{2})$, e.g. $\sigma = 10^{-4}$. It should be remarked here that the performance of the algorithm is insensitive to the choice of these two parameters. Under exact arithmetic, the algorithm guarantees convergence to the unique optimal solution $z^\star$ in a finite number of iterations. In practice the algorithm can be stopped as soon as

$$\left\| z^k - \left[z^k - \gamma\left(Qz^k + q\right)\right]_{\underline{z}}^{\overline{z}} \right\| \le \gamma\sqrt{2\mu\epsilon},$$

where μ is (lower bound on) the smallest eigenvalue of the positive definite matrix Q, for some given error tolerance $\epsilon > 0$. This condition guarantees that $V(\hat{z}^k, x) - V^* \le \epsilon$ for $\hat{z}^k = \left[z^k - \gamma\left(Qz^k + q\right)\right]_{\underline{z}}^{\overline{z}}$, where V^* is the optimal value of problem (2), respectively.

4.2 Fixed-point basics

By the term *fixed-point* we denote a way to represent numbers on digital calculators, meaning a norm to map real numbers into a sequence of binary values. Fixed-point representations are of particular interest in embedded applications as they grant fast and efficient additions and multiplications on nearly all computing devices.

A fixed-point data type is characterized by four parameters: (1) the total word length w, (2) the signedness s, (3) the number r of bits for the integer part, and (4) the number p of bits for the fractional part.

A round-off error may occur when converting a number $\zeta \in \mathbb{R}$ into its fixed-point representation $\texttt{fi}(\zeta)$. This error is bounded by a function of the number of fractional bits as follows: $|\zeta - \texttt{fi}(\zeta)| \le 2^{-(p+1)}$. The same rounding error occurs when a multiplication is performed; extending this result, it is possible to bound the total error for an inner product operation as follows

$$|x'y - \texttt{fi}(x'y)| \le 2^{-(p+1)}n, \tag{3}$$

Algorithm 1 Proximal Newton algorithm for box-constrained QP

Input: Q, q, $\underline{z}$, $\overline{z}$, $\sigma \in (0, \frac{1}{2})$, $\gamma < 1/\lambda_{\max}(Q)$, $z^0 \in \mathbb{R}^n$

1: **for** $k = 1, 2, \ldots$ **do**
compute Newton direction
2: $\beta \leftarrow \left\{ i | \underline{z}_i < z_i^k - \left(Q_i z^k + q_i\right) < \overline{z}_i \right\}$
3: $d_i \leftarrow -\left(z_i - \left[z_i^k - \gamma \left(Q_i z^k + q_i \right) \right]_{\underline{z}_i}^{\overline{z}_i} \right)$, $i \notin \beta$
4: $d_\beta \leftarrow -Q_{\beta\beta}^{-1} \left(Q_\beta z_\beta^k + q_\beta + Q_{\beta\neg\beta} d_{\neg\beta} \right)$
perform line search
5: $\alpha \leftarrow 1$
6: **while** $F_\gamma\left(z^k + \alpha d\right) - F_\gamma\left(z^k\right) > \sigma\alpha\nabla F_\gamma\left(z^k\right)' d$ **do**
7: $\alpha \leftarrow \frac{\alpha}{2}$
8: **end while**
9: $z^{k+1} \leftarrow z^k + \alpha d$, $k \leftarrow k + 1$
10: **end for**

Output: z^*

where $x, y \in \mathbb{R}^n$, and for a matrix-vector product,

$$\|Ax - \texttt{fi}(Ax)\|_\infty \leq 2^{-(p+1)} n. \tag{4}$$

Moreover, numbers can be represented in a limited range dependent on the number of integer bits, $[-2^r, 2^r - 1]$. Values exceeding these bounds cause an overflow or underflow error and, depending on the architecture, are either saturated or cast back into the representable space (*modulo* arithmetic). Either case must be avoided, since can critically affect the convergence of Algorithm 1.

4.3 Round-off error analysis

We now analyze the effects of round-off errors occurring when executing Algorithm 1 supported by a fixed-point number representation with p bits for the fractional part. Assume that input data is represented exactly, i.e. $\texttt{fi}(x) = x$ for $x = \{Q, q, \underline{z}, \overline{z}, \epsilon, \sigma\}$. The goal is to bound the round-off error accumulated on the optimization vector z during the execution of one algorithm iteration.

We proceed by bounding the error on the Newton direction $\texttt{fi}(d) - d$ when executing lines (2-4). The values of d are updated either according to line 3, by computing a matrix-vector product and an Euclidean projection, or according to line 4, where a matrix-vector product and the solution of a linear system is required. Computing the projection does not cause additional errors; on the other hand (as detailed shortly) this happens for the linear system. Therefore, we analyze the error on d by considering the worst-case scenario where $\beta = \{1, 2, \ldots, Nn_u\}$ and d is updated by solving a linear system whose dimension is equal to the number of variables of Problem (2).

From standard linear algebra results (see, e.g., van der Sluis (1970); Wilkinson (1994)), given a perturbed linear system in the form $Q\left(x + \tilde{x}\right) = b + \tilde{b}$, it holds that

$$\begin{aligned} \|\tilde{x}\| &\leq \kappa(Q) \frac{\|\tilde{b}\|}{\|b\|} \|x\| \\ &\leq \kappa(Q) \|\tilde{b}\| \|Q^{-1}\|, \end{aligned} \tag{5}$$

where $\kappa(Q) = \|Q\|\|Q^{-1}\|$ is the condition number of matrix Q. In our case $b = Qz^k + q$ and $\tilde{x} = d - \texttt{fi}\left(d\right)$. Substituting into (5) and taking into account bound (4) we obtain

$$\|d - \texttt{fi}\left(d\right)\| \leq \kappa(Q) \|Q^{-1}\| 2^{-(p+1)} \left(Nn_u\right)^{3/2} \tag{6}$$

The line search operation does not cause round-off errors to accumulate on z^k, since it only involves halving the stepsize α. Finally, in line (9) z^k acquires the perturbation on d, bounded by (6), plus a final round-off due to multiplication by α. Therefore, the bound on round-off errors accumulated on the optimization vector in one iteration becomes

$$\|z - \texttt{fi}\left(z\right)\| \leq \left(1 + \kappa(Q) \|Q^{-1}\| \left(Nn_u\right)^{3/2}\right) 2^{-(p+1)} \tag{7}$$

4.4 Avoiding overflow errors

Overflow errors occur when trying to store a number outside of the representable range $[-2^r, 2^r - 1]$; to avoid them, r must be chosen large enough such that every computed value during the execution of the algorithm lies within the admissible range. Provided that r is chosen large enough to represent all static problem data, we now give lower bounds for it that guarantee the representability of variable data as well, namely z and d vectors.

Let ε_d and ε_z be the right-hand sides of inequalities (6) and (7), respectively. Then,

$$\begin{aligned} \|\texttt{fi}\left(z\right)\|_\infty &= \|\texttt{fi}\left(z\right) - z + z\|_\infty \\ &\leq \|z\|_\infty + \|\texttt{fi}\left(z\right) - z\|_\infty \\ &\leq \max\left\{\|\underline{z}\|_\infty, \|\overline{z}\|_\infty\right\} + \varepsilon_z \triangleq \hat{z}. \end{aligned} \tag{8}$$

Moreover,

$$\begin{aligned} \|\texttt{fi}\left(d\right)\|_\infty &= \|\texttt{fi}\left(d\right) - d + d\|_\infty \\ &\leq \|d\|_\infty + \|\texttt{fi}\left(d\right) - d\|_\infty \\ &\leq \|Q^{-1}\|_\infty \left(\|Q\|_\infty \hat{z} + \|q\|_\infty\right) + \varepsilon_d \triangleq \hat{d}. \end{aligned} \tag{9}$$

Therefore, the execution of Algorithm 1 supported by fixed-point number representation with r bits for the integer part does not cause overflow or underflow errors if

$$r \geq \log_2\left(\max\left\{\hat{z}, \hat{d}\right\} + 1\right) + 1, \tag{10}$$

5. OPTIMIZATION OF THE ALGORITHM

In this section we analyze key issues regarding algorithm implementation in fixed-point arithmetic and propose procedures to deal with them and optimize computation efficiency.

5.1 Preconditioning

The accuracy of the result when computing the solution of a perturbed linear system is sensible to the *conditioning* of the problem, as shown in (5). This sensitivity is directly reflected in the accuracy of the overall solution of the QP given by Algorithm 1, where an ill-conditioned Hessian cause a significant degeneration of its performance. The impact of this phenomenon can however be reduced by *preconditioning* the problem, i.e. finding a change of coordinates $\bar{z} = \bar{P}^{-1} z$, such that the condition number of the Hessian of the transformed problem, $\bar{P}Q\bar{P}$, is smaller than that of of (2), Q. Ideally $\bar{P}$ should solve the problem

$$\begin{aligned} \text{minimize} \quad & \kappa\left(PQP\right) \\ \text{subject to} \quad & P \in \mathbb{R}^{n \times n} \text{ is PSD} \end{aligned} \tag{11}$$

In general, the preconditioner resulting from (11) will not preserve the box structure of the feasible set $\mathcal{Z}$. This can be avoided by forcing P diagonal. Solving problem (11),

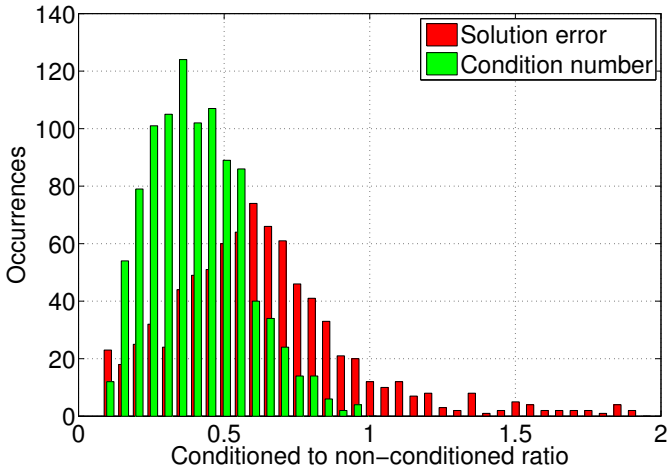

Fig. 1. Impact of preconditioning on the Hessian condition number and the solution relative error.

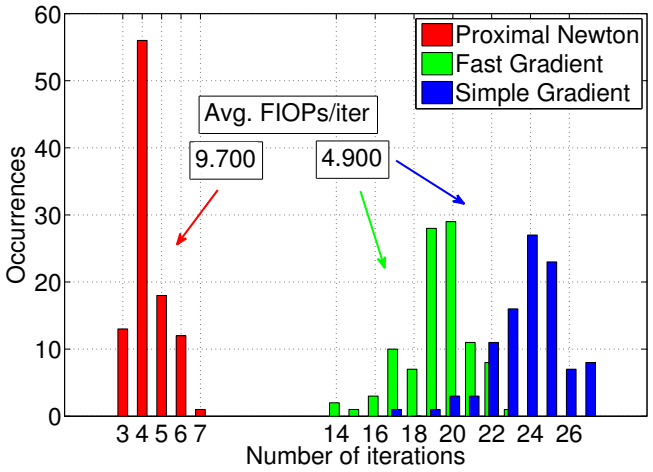

Fig. 2. Proximal Newton method compared to Gradient methods for required number of iterations in fixed-point arithmetic and average fixed-point computations (FIOPs) per iteration.

as shown by Boyd (1994), is equivalent to solving the following generalized eigenvalue problem (GEVP)

$$\begin{aligned} &\text{minimize} && \xi \\ &\text{subject to} && S > 0,\ S \text{ diagonal},\ \xi \geq 0 \\ & && S \leq Q \leq \xi S \end{aligned} \tag{12}$$

and picking $P = S^{-1/2}$. In order to avoid the inclusion of a bilinear matrix inequality constraint, problem (12) can be reformulated (for an alternative approach, see Richter et al. (2012)) by means of the Schur complement as follows

$$\begin{aligned} &\text{maximize} && \delta \\ &\text{subject to} && S > 0,\ S \text{ diagonal},\ \delta \geq 0 \\ & && \begin{bmatrix} S & \delta I \\ \delta I & Q^{-1} \end{bmatrix} \geq 0 \end{aligned} \tag{13}$$

Figure 1 highlights the effects of preconditioning on 1000 random QP problems of variable sizes in the range $[20, 200]$ and variable Hessian condition numbers. The green histogram plots the variation of the condition number expressed as $\kappa(PQP))/\kappa(Q)$. The red histogram plots the variation of the solution errors expressed as $\|\zeta^* - z^C\|/\|z^* - z^C\|$, where ζ^* is the solution of the preconditioned problem and z^C is the solution given by the solver of *CPLEX* (`http://www.ibm.com/software/commerce/optimization/cplex-optimizer/`). Results show an average 56.4 % reduction of condition number of the Hessian and 34.7 % of the solution error. Note that preconditioning can be computed offline. However, preconditioning is not beneficial for all the problems: in 10.6 % of the cases the solution error is increased. This happens since a modification of the Hessian may cause a change in the constraint activation behavior during algorithm execution, and therefore the size of the linear system that has to be solved at each iteration.

5.2 Division-free computations

An efficient way to solve the linear system in step 4 of Algorithm 1 is by means of a Cholesky factorization followed by forward and backward substitution. However, these procedures require the computation of the reciprocal and the square root of the diagonal entries of the matrix. Performing a division on most embedded devices requires more cycles than performing additions and multiplications; therefore, the presence of divisions can cause a significant degeneration of overall performances.

A possible approach for division-free computations is to scale the QP problem such that the Hessian has all entries in the range $[-1, 1]$; this means (since it is SPD) that all diagonal entries fall in the range $(0, 1]$. The subsequent step is to store into the device a pre-computed look-up table containing $1/\sqrt{\xi}$ with ξ covering all the fixed-point values in the range $(0, 1]$ for the selected precision; then, to evaluate the inverse of a desired value it is sufficient to access the table with the value itself as index, and square the result.

This solution constitutes a trade-off knob between computation speed and memory occupancy, increasing the latter by $w \cdot 2^p/8$ bytes.

6. SIMULATIONS

6.1 Computational complexity

We first compare the computational complexity for the implementation in fixed-point arithmetic of Algorithm 1 against gradient-based methods (see Nesterov (1983, 2004)). The reason of this choice is due to the interest arising recently in embedded implementations of first-order algorithms, which have been proven to perform well on low-precision arithmetic (cf. Patrinos et al. (2013) for a dual gradient method or Jerez et al. (2013) for a primal, accelerated version).

Figure 2 shows the histogram distribution of the number of iterations required to reach a target solution quality. Computations are performed in fixed-point arithmetic with word length $w = 32$ bits and fraction length $p = 16$ bits. Results are based on 100 random QPs of size $n = 50$ and show that iteration count lies in the range $[3, 7]$ for proximal Newton, $[14, 23]$ for fast gradient, and $[17, 27]$ for simple gradient methods. However, the estimated average fixed-point operations (multiplications and additions) performed per-iteration is roughly double for the Newton method compared to gradient-based ones.

Table 1 shows how per-iteration and overall computation complexities scale with number of variables n. The goal is to estimate the exponent ρ when fitting the actual fixed-point operations as a function of the problem size, according to the relation $a \cdot n^\rho$. The theoretical per-iteration complexity bound for the gradient methods is $O(n^2)$, due to the computation of a matrix-vector product, and this

Table 1. Estimation of the exponential coefficient for single iteration (ρ_i) and overall algorithm (ρ_A) complexities with respect to problem size (with 95% confidence bounds).

Method	ρ_i	ρ_A
Proximal Newton	2.32 ± 0.08	2.544 ± 0.12
Fast Gradient	2.01 ± 0.01	2.19 ± 0.09
Simple Gradient	2.01 ± 0.01	2.21 ± 0.12

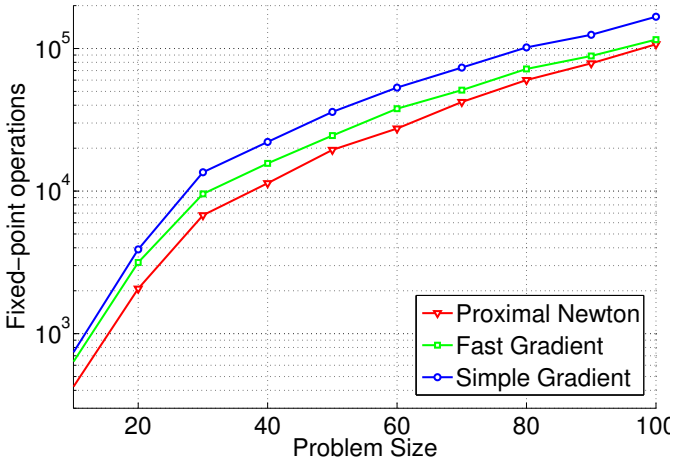

Fig. 3. Proximal Newton method compared to Gradient methods for overall number of fixed-point operations varying problem size.

is confirmed by the simulations. Newton-based methods are instead bounded by $O(n^3)$ due to the solution of a linear system; however, the effective complexity growth is estimated as $n^{2.32}$. This happens because the proposed algorithm requires only the solution of a linear system of reduced order.

Finally, Figure 3 shows a comparison of the overall fixed-point operations executed for matrix computations, estimated over 200 random QPs of different sizes ranging from $n = 10$ to $n = 100$ variables. Results show that the proposed implementation is indeed computationally efficient; however, the benefits compared to gradient methods decrease for larger problems, and eventually vanish due to the higher exponential dependency on n (cf. Table 1).

6.2 Solution accuracy

The following simulation shows how solution accuracy varies with the number of fractional bits chosen in the fixed-point representation, comparing Algorithm 1 (Newton) with fast and simple gradient methods (FGM and GM, respectively). By 'solution accuracy' we mean the relative discrepancy ϵ_z with the solution z^C obtained from the state-of-the-art solver of *CPLEX* running on double-precision arithmetic, that is

$$\epsilon_z = \frac{\|z^* - z^C\|}{\|z^C\|}. \qquad (14)$$

Figure 4 shows average solution errors, computed as in (14), over 100 QPs of size 10 and 100 QPs of size 120, varying the number of fractional bits from $p = 4$ to $p = 16$. In compliance with the bound in (7) on the round-off error accumulation, we observe an exponential decrease with respect to p. Results show a remarkable robustness of the proposed implementation when running in finite-precision arithmetic. Nevertheless, it is more susceptible

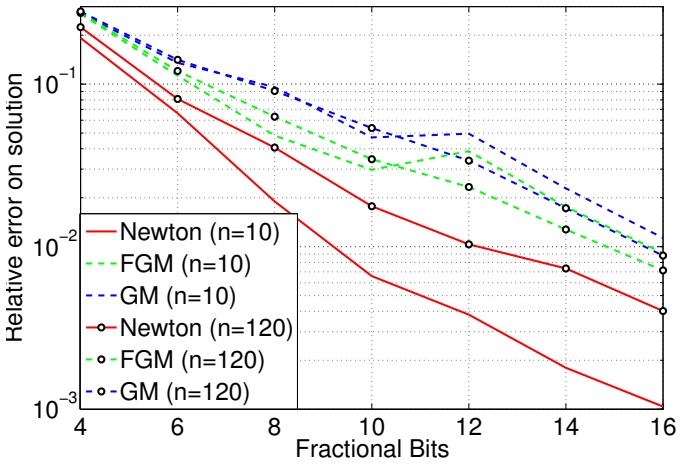

Fig. 4. Achieved solution accuracy with proximal Newton and gradient methods for two problem sizes, varying number of fractional bits.

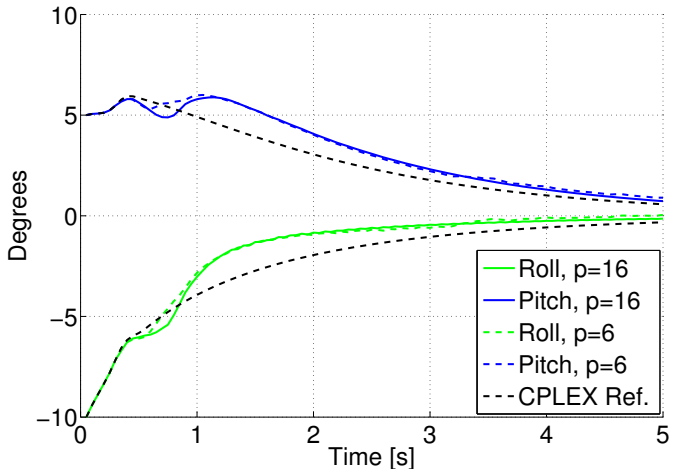

Fig. 5. AFTI-F16. Closed-loop trajectories obtained feeding proximal Newton optimal inputs for different fixed-point precisions, compared to double-precision reference solver of *CPLEX*

to variations on problem size, as reflected by the term $(Nn_u)^{3/2}$ in (7).

6.3 Control of a F16 aircraft

We verify the closed-loop behavior of a simulated physical system connected to a predictive controller that relies on an implementation in fixed-point arithmetic of Algorithm 1 to solve on-line the QP problem. The goal is to regulate roll and pitch angles of an AFTI-F16 aircraft.

The controller was based on a linearized aircraft model of Kapasouris et al. (1988) consisting of four states (namely roll, pitch, yaw and attack angles) and two inputs (elevator and flaperon angles). Both inputs are constrained in the $[-25°, +25°]$ range. The system is open-loop unstable.

Figure 5 shows the closed-loop trajectories of roll and pitch angles. Two implementations with fixed-point arithmetic of Algorithm 1, with 16 and 6 bits for the fractional part, are compared with the trajectories obtained acquiring control inputs from the state-of-the-art solver of *CPLEX*, running in double-precision arithmetic. Results show that a predictive controller supported by the proposed implementation with fixed-point arithmetic is able to stabilize

Table 2. Hardware implementation

Vars.	T_{fi} $[ms]$	T_{fl} $[ms]$	$Size_{fi}$ $[KB]$	$Size_{fl}$ $[KB]$
10	0.6	1.8	14.7	20.2
20	1.8	5	16	22
30	4.7	9.7	18	27
40	8.8	45.5	21	32.8
50	14.7	89.8	24.8	39.6
60	25.7	100.8	29.3	59.2
70	44.8	n/a	34.6	n/a
80	52.4	n/a	40.7	n/a

the system. Note that the controller sampling time should be fast enough such that oscillations in system inputs due to the quantization of the QP solution fall outside the process bandwidth.

7. HARDWARE IMPLEMENTATION

An implementation of Algorithm 1 was deployed on a low-power, low-cost, ARM-based Cortex-M3 general-purpose processing unit, model Atmel SAM3X8E. This unit features a 32-bit CPU operating at 84 MHz, 100 KB of RAM and 512 KB of flash memory.

The device was assigned to solve a set of random QPs of increasing size. The algorithm was coded both with floating-point arithmetic (word length of 32 bits) and fixed-point arithmetic (word length of 16 bits, of which 8 bits for the fractional part).

Table 2 shows the average experimental results: for each problem size ranging from 10 to 80 variables, we report overall computation time for the fixed-point arithmetic (T_{fi}) and floating-point arithmetic (T_{fl}) versions; similarly, we report code size for the compiled binary.

Table 2 shows that switching from floating- to fixed-point arithmetic causes the computation time and code size to become up to 4 and 2 times smaller, respectively. Advantages become more evident as the number of variables increases; for problems with $n \geq 70$ the floating point version is not able to converge at all, probably due to lack of memory.

8. CONCLUSION

In this work we investigated in detail the implementation with fixed-point arithmetic of a Newton-based QP solver for embedded MPC applications. We analyzed the propagation of the round-off error coming from the quantization of the number representation space, and derived design guidelines to avoid overflow errors. Simulation results showed a good algorithm performance and solution accuracy when compared to gradient-based approaches. For ill-conditioned problems, we proposed an optimal scaling method to hinder them. Finally, we showed how the proposed solver can be deployed in low-cost, low-power hardware.

REFERENCES

A. Bemporad. Model predictive control design: New trends and tools. In *Proc. 45th IEEE Conference on Decision and Control*, 2006.

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 2002.

A. Bemporad, A. Oliveri, T. Poggi, and M. Storace. Ultra-Fast Stabilizing Model Predictive Control via Canonical Piecewise Affine Approximations. *Automatic Control, IEEE Transactions on*, 56(12):2883–2897, 2011.

S. Boyd. *Linear matrix inequalities in system and control theory*, volume 15. Siam, 1994.

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. of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 209–218. ACM, 2011.

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 Conference*, 2013.

P. Kapasouris, M. Athans, and G. Stein. Design of feedback control systems for unstable plants with saturating actuators. *NASA STI/Recon Technical Report N*, 89: 14377, 1988.

E. C. Kerrigan, J. L. Jerez, S. Longo, and G. A. Constantinides. Number representation in predictive control. In *Proc. IFAC Conference Nonlinear Model Predictive Control*, pages 60–67, 2012.

G. Knagge, A. Wills, and A. Mills. ASIC and FPGA implementation strategies for model predictive control. *Proc. European Control Conference*, 2009.

K. V. Ling, B. F. Wu, and J. M. Maciejowski. Embedded model predictive control (MPC) using a FPGA. *Proc. 17th IFAC World Congress*, 2008.

S. Longo, E. C. Kerrigan, and G. A. Constantinides. A predictive control solver for low-precision data representation. In *Proc. 51st IEEE Conference on Decision and Control*, 2012.

J. M. Maciejowski. *Predictive Control: With Constraints*. Prentice Hall, 2002.

Y. Nesterov. A method of solving a convex programming problem with convergence rate O (1/k2). *Soviet Mathematics Doklady*, 1983.

Y. Nesterov. *Introductory lectures on convex optimization: A basic course*. 2004.

P. Patrinos and A. Bemporad. Proximal Newton methods for convex composite optimization. In *Proc. 52st IEEE Conference on Decision and Control*, pages 2358–2363, 2013.

P. Patrinos and H. Sarimveis. A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings. *Automatica*, 46(9):1405–1418, 2010.

P. Patrinos, A. Guiggiani, and A. Bemporad. Fixed-point dual gradient projection for embedded model predictive control. In *Proc. 12th European Control Conference*, pages 3602–3607, 2013.

S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 2003.

J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, 2009.

S. Richter, C. N. Jones, and M. Morari. Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method. *Automatic Control, IEEE Transactions on*, 57(6):1391–1403, 2012.

A. van der Sluis. Stability of solutions of linear algebraic systems. *Numerische Mathematik*, 14(3):246–251, 1970.

P. D. Vouzis, M. V. Kothare, L. G. Bleris, and M. G. Arnold. A System-on-a-Chip Implementation for Embedded Real-Time Model Predictive Control. *Control Systems Technology, IEEE Transactions on*, 17(5): 1006–1017, 2009.

J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Dover, 1994.

A. G. Wills, G. Knagge, and B. Ninness. Fast Linear Model Predictive Control Via Custom Integrated Circuit Architecture. *Control Systems Technology, IEEE Transactions on*, 20(1):59–71, January 2012.