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

Embedded Predictive Control on an FPGA using the Fast Gradient Method

Juan L. Jerez, Paul J. Goulart, Stefan Richter, George A. Constantinides, Eric C. Kerrigan and Manfred Morari

***Abstract*— Model predictive control (MPC) in resource-constrained embedded platforms requires faster, cheaper and more power-efficient solvers for convex programs than is currently offered by software-based solutions. In this paper we present the first field programmable gate array (FPGA) implementation of a fast gradient solver for linear-quadratic MPC problems with input constraints. We use fixed-point arithmetic to exploit the characteristics of the computing platform and provide analytical guarantees ensuring no overflow errors occur during operation. We further prove that the arithmetic errors due to round-off can lead only to reduced accuracy, but not instability, of the fast gradient method. The results are demonstrated on a model of an industrial atomic force microscope (AFM) where we show that, on a low-end FPGA, satisfactory control performance at a sample rate beyond 1 MHz is achievable, opening up new possibilities for the application of MPC.**

I. Introduction

Model predictive control (MPC) provides a systematic approach for handling physical constraints in control systems, often leading to improved control performance and reduced tuning effort for new applications. However, the intense computational demands made by MPC methods preclude their use in applications that could benefit considerably from its advantages, especially those that run on resource-constrained, embedded computing platforms.

For linearly constrained MPC problems of low dimensionality, one can partially avoid this computational burden by precomputing the solution map offline [3]. For larger problems, this approach quickly becomes impractical, mainly due to substantial memory requirements, forcing a return to online optimization methods.

Recently, there has been significant interest in Nesterov's fast gradient method (FGM) for the online solution of linear-quadratic MPC problems [6], [11], [18]–[20]. Compared to active-set or interior-point schemes, the fast gradient method does 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, make the fast gradient method a promising candidate for efficient, low cost MPC.

Juan L. Jerez and George A. Constantinides are with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ, United Kingdom, jlj05|gac1@imperial.ac.uk

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

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

There have been several recent efforts to develop customized optimization solvers for MPC problems. In terms of software, [5], [14] describe automatic generators of interior-point solvers. For hardware, [8], [22], [24] report different custom computing architectures for both interior-point and active-set methods using floating-point arithmetic in FPGAs. Although there has been some progress in accelerating the core component of these algorithms – solvers for linear equations – using fixed-point arithmetic [9], extending these results to the other aspects of interior-point or active-set algorithms remains challenging.

In this paper, we present a parameterized automatic generator of custom hardware computing architectures for the FGM and derive theoretical results that guarantee both the absence of overflow and bounded error accumulation due to round-off errors. As a proof of concept, a generated solver instance is demonstrated for an input-constrained, linear-quadratic MPC problem for a real-world, highly dynamic positioning system inside an AFM, running at a controller sampling rate in excess of 1MHz.

After a brief summary of input-constrained MPC and the FGM in Sections II and III, we focus on the fixed-point aspects of a fast gradient implementation in Section IV. Since it is crucial to prevent overflow errors for a reliable operation in fixed-point arithmetic, we provide results that ensure absence of such errors for all iterations of the method. Furthermore, we present a convergence analysis under (inexact) fixed-point computations. This analysis underpins the numerical stability of the method for hardware implementations and can be used to choose the minimum number of bits so that reliable operation is preserved at minimal resource usage. In Section V, we describe the solver fixed-point hardware design for low cost and high-speed embedded optimization implementations. Finally, in Section VI, the contributions are verified for the AFM MPC. The proposed hardware-based implementation of the MPC controller achieves solution times below 1 µs, enabling operation at the required sampling rate of more than 1 MHz on a low-end FPGA.

II. Input-Constrained MPC

At every sampling instant, given an estimate or measurement of the current state of the plant x, an MPC-based controller for an input-constrained system solves a finite-horizon optimal control problem in the form

$$\begin{aligned} \min \; & \frac{1}{2}x_N^T Q_N x_N + \sum_{k=0}^{N-1}\left[\frac{1}{2}\left(x_k^T Q x_k + u_k^T R u_k\right) + x_k^T S u_k\right] \quad (1)\\ \text{s.t.} \; & x_0 = x, \\ & x_{k+1} = A_d x_k + B_d u_k, \quad \text{for } k = 0,1,\ldots,N-1, \\ & u_k \in \mathbb{U}, \quad \text{for } k = 0,1,\ldots,N-1, \end{aligned}$$

with state $x_k \in \mathbb{R}^{n_x}$ and input $u_k \in \mathbb{R}^{n_u}$ confined to the convex, compact set $\mathbb{U}$ containing the origin in its interior. We assume throughout that the penalty matrix $Q \in \mathbb{R}^{n_x \times n_x}$ is 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 (1) is jointly convex in the states and inputs. It is well-known that a receding horizon implementation of the optimal solutions to this optimization problem produces a stable feedback controller, provided that Q_N is chosen appropriately and N is sufficiently large [15], even when no terminal state constraints are imposed.

For the sake of simplicity, all of the results of this paper are presented with reference to the optimal control problem in regulator form in (1). However, these results are easily generalized to setpoint tracking problems.

III. 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 [16]. The method can be applied to the solution of MPC problem (1) if the states in the problem formulation are eliminated by expressing them as a function of the current state x and the future input sequence (called *condensing* [13]), resulting in the problem

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

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

In this paper we consider the *constant step scheme II* of the FGM in [17, §2.2.3]. Its algorithmic scheme for the solution of (2), optimized for parallel execution on parallel hardware, is given in Algorithm 1. Note that the state-independent terms $(I - \frac{1}{L}H)$, $\frac{1}{L}\Phi$ and $(1+\beta)$ can all be computed offline and 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 in MPC the set $\mathbb{K}$ is the direct product of N n_u-dimensional sets, it suffices to consider N independent projections that can be performed independently. For the most common case of a box constraint on the control input, every single 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 [17, 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

$$\min\left\{\left(1-\sqrt{\frac{1}{\kappa}}\right)^i, \frac{4\kappa}{(2\sqrt{\kappa}+i)^2}\right\} \cdot 2\big(f(z_0;x) - f^*(x)\big), \quad (3)$$

for all $i = 0, \ldots, I_{\max}$, where κ denotes the condition number of f, or an upper bound of it, given by $\kappa := L/\mu$,

Algorithm 1 Fast gradient method for the solution of MPC problem (2) 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, step size $\beta := \big(\sqrt{L} - \sqrt{\mu}\big)/\big(\sqrt{L} + \sqrt{\mu}\big)$

1: **for** $i = 0$ to $I_{\max} - 1$ **do**
2: $\quad t_i := (I - \frac{1}{L}H)y_i - \frac{1}{L}\Phi x$
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**
6: **return** $z_{I_{\max}}$

where L and μ are a Lipschitz constant for the gradient of f and a convexity parameter of f, respectively. Based on this convergence result, which states that the bound exhibits the best of a linear and a sublinear rate, one can derive a certified, practically relevant iteration bound $I_{\max}$ such that the final residual is guaranteed to be within a specified level of suboptimality for all states from a bounded set [19].

The inclusion of state constraints in the MPC problem (1) would change the geometry of the feasible set and make the projection subproblem as hard as the original problem, since the constraints would no longer be separable in the inputs u_k. A standard workaround is to solve the MPC problem in the dual domain, again using the fast gradient method [20].

IV. Fixed-Point Aspects

We will first motivate the use of fixed-point arithmetic from a hardware perspective in Section IV-A and then isolate potential error sources under this arithmetic in Section IV-B. We concentrate on two types of errors. For *overflow errors* we provide analysis to guarantee that they cannot occur (Section IV-D), where for *arithmetic round-off errors* we prove that there is a converging upper bound on the total incurred error (Section IV-E). The results we obtain hold under the assumptions in Section IV-C and guarantee reliable operation of the FGM on fixed-point platforms.

A. Fixed-Point vs. Floating-Point Arithmetic

General purpose 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. In this case, *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, higher power consumption and longer arithmetic delays.

In contrast, *fixed-point* numbers have a fixed number of bits for the integer and fraction fields, i.e. the exponent does

not vary and does not have 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 acceleration through extra parallelization in a custom hardware implementation. For instance, in a Xilinx FPGA, fixed-point addition takes one 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 [1].

B. Error Sources in Fixed-Point Arithmetic

The benefits of fixed-point arithmetic motivate its use in the FGM to realize fast and efficient implementations 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, there are several types of errors that must be accounted for:

Representation Errors: Representation errors arise when converting the problem and algorithm data, e.g. Hessian H or step size β, from floating- to fixed-point numbers. Potential consequences include the loss of 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, addition and subtraction involve no round-off error in fixed-point arithmetic, provided the result has the same number of fraction bits as the operands [23]. 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}. Note that in 2's complement arithmetic, truncation incurs a negative error both for positive and negative numbers.

C. Problem Setup and Assumptions for the Error Analysis

We henceforth consider the solution to the following normalized fixed-point representation of the optimization problem (2):

$$f_n^*(\hat{x}) := \min_z f_n(z;\hat{x}) := \frac{1}{2} z^T \hat{H}_n z + z^T \hat{h}_n(\hat{x}) \quad \text{subject to } z \in \widehat{\mathbb{K}}, \tag{4}$$

whose problem data will be explained below. We use $\hat{(\cdot)}$ throughout in order to distinguish variables in fixed-point arithmetic from those in exact arithmetic. We assume the following on the problem and algorithm data:

Assumption 1: All variables are represented using the same number of fraction bits b.

Assumption 2: The feasible set under finite precision, $\widehat{\mathbb{K}} := \left\{ z \in \mathbb{R}^n \,\middle|\, \hat{z}_{\min} \le z \le \hat{z}_{\max} \right\}$, is an n-dimensional box and is contained in the original set, i.e. $\widehat{\mathbb{K}} \subseteq \mathbb{K}$.[1]

[1]Since the truncation errors are negative, one has to contract the original lower bound $z_{\min}$ of set $\mathbb{K}$ by 2^{-b} to avoid the possibility of the fixed-point implementation returning infeasible solutions w.r.t. the original problem.

Assumption 3: Let $\hat{H}$ be the fixed-point representation of Hessian H and $\lambda_{\max}(\hat{H})$ its maximum eigenvalue (in exact arithmetic). The number of fraction bits b and/or the constant $c \ge 1$ are chosen such that the fixed-point representation of

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

denoted by $\hat{H}_n$, has

(i) a maximum eigenvalue of less than or equal to one
(ii) and a minimum eigenvalue that is greater than zero.

Assumption 4: Given the number of fraction bits b and constant c determined according to Assumption 3, we denote by $\hat{\Phi}_n$ the fixed-point representation of

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

Furthermore, we use $\hat{h}_n(\hat{x})$ to denote the result of the matrix-vector product $\hat{\Phi}_n \hat{x}$ in *fixed-point* arithmetic.

Assumption 5: The fixed-point step size $\hat{\beta}$ is chosen such that

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

Assumption 1 will allow for a simplified analysis in Section IV-E, whereas Assumption 2 ensures feasibility of the (approximate) solution returned by the fast gradient method. Assumption 3 makes sure that the objective function in (4) remains strongly convex and Assumption 5 guarantees that the *true* condition number of $\hat{H}_n$ is not underestimated, in which case the convergence result of the FGM in (3) would be invalid. In fact, the assumption ensures that the effective condition number for the convergence result is given by

$$\kappa_n = \left(\frac{1+\hat{\beta}}{1-\hat{\beta}}\right)^2 \ge \kappa(\hat{H}_n). \tag{5}$$

D. Preventing Overflow Errors

In order to avoid overflow errors in a fixed-point implementation of Algorithm 1 (adapted for problem (4)), the largest absolute values of the iterates' and intermediate variables' components must be known or upper-bounded *a priori* to decide how many bits to allocate for their *integer parts*. For the *static* problem data $(I - \hat{H}_n)$, $\hat{\Phi}_n$, $1 + \hat{\beta}$ and $\hat{\beta}$, the number of integer bits is easily determined by the maximum value in each expression. For the *dynamical* data, which changes in every iteration of the fast gradient method, we summarize the upper bounds in the following proposition.

Proposition 1: If problem (4) 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

$$\begin{aligned}
\|\hat{z}_{i+1}\|_\infty &\le \bar{z} := \max\left\{\|\hat{z}_{\min}\|_\infty, \|\hat{z}_{\max}\|_\infty\right\}, \\
\|\hat{y}_{i+1}\|_\infty &\le \bar{y} := \bar{z} + \hat{\beta}\|\hat{z}_{\max} - \hat{z}_{\min}\|_\infty, \\
\|(I - \hat{H}_n)\,\hat{y}_i\|_\infty &\le \bar{y}_{\text{inter}} := \|I - \hat{H}_n\|_\infty \cdot \bar{y}, \\
\|\hat{x}\|_\infty &\le \bar{x} := \max_{x \in \widehat{\mathbb{X}}_0} \|x\|_\infty, \\
\|\hat{\Phi}_n \hat{x}\|_\infty &\le \bar{h} := \|\hat{\Phi}_n\|_\infty \cdot \bar{x}, \text{ and} \\
\|t_i\|_\infty &\le \bar{t} := \bar{y}_{\text{inter}} + \bar{h},
\end{aligned} \tag{6}$$

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. ∎

Normalization of the objective as introduced in Section IV-C has no effect on the maximum absolute values of the iterates. Furthermore, the bound in (6) also applies for the intermediate elements/cumulative sums in the evaluation of the matrix-vector product. Notice as well that most of the bounds stated in Proposition 1 are indeed tight bounds.

E. Stability Analysis Under Arithmetic Round-Off Errors

We show next how to derive an upper bound on the sequence of residuals $\{f_n(\hat{z}_i;\hat{x}) - f_n^*(\hat{x})\}_{i=0}^{I_{\max}}$, where $\hat{z}_i$ is a *fixed-point* iterate of the fast gradient method for the solution of problem (4) and $\hat{x}$ is the *fixed-point* representation of the initial state. For this purpose, we relate iterates $\hat{z}_i$ to the iterates generated under *exact arithmetic*, z_i, which then leads to the bound

$$f_n(\hat{z}_i;\hat{x}) - f_n^*(\hat{x}) \leq f_n(z_i;\hat{x}) - f_n^*(\hat{x}) + \Delta_i(\hat{x}), \quad (7)$$

where for an increasing number of iterations $i \to +\infty$:

- the residual with respect to the iterates in exact arithmetic, $f_n(z_i;\hat{x}) - f_n^*(\hat{x})$, converges to zero according to (3) (note that the condition number in case of the fixed-point implementation is given by κ_n in (5)),
- and the term $\Delta_i(\hat{x})$, which is due to arithmetic round-off errors, converges to some *positive* constant $\Delta_\infty(\hat{x})$.

Consequently, although there is a persistent worst case error in the residual due to arithmetic round-off errors, this result ensures that the error does not accumulate continually, which is a crucial prerequisite for reliable operation of the fast gradient method on fixed-point platforms.

In the literature, there are different approaches for a convergence analysis of the FGM under inexact computations. However, none of them address errors from fixed-point arithmetic explicitly. Existing research can be classified into

(i) approaches that derive conditions to guarantee that no continual error accumulation occurs, e.g. [2], [21], and
(ii) an approach that investigates convergence in a general error framework [4], concluding that errors can be accumulated, which leads to divergence of the method.

Although arithmetic round-off errors due to fixed-point arithmetic could be interpreted within the framework of [2] in order to prove a converging upper bound $\Delta_i(\hat{x})$, we provide a complimentary and much shorter analysis based on control theory for the most common case of strongly convex quadratic objectives.[2]

The following proposition provides the entry point for the numerical stability analysis:

Proposition 2: Assume that problem (4) is solved by the FGM using the appropriately adapted Algorithm 1. Let b be the number of fraction bits and n be the dimension of the decision vector. *Any* accumulation of finite precision errors up to iteration i, $\eta_i := \hat{z}_i - z_i$, $i = 0, \ldots, I_{\max}$, can be described by the two-step recurrence

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

for $\eta_{-1} = \eta_0$ and *some* arithmetic round-off errors due to

- matrix-vector multiplication (line 2), $\epsilon_{t,i} \in [-n2^{-b}, 0]^n$,
- scalar-vector multiplication (line 4), $\epsilon_{y,i} \in [-2^{-b}, 2^{-b}]^n$, with $\epsilon_{y,-1} = 0$,

and *some* vectors $\epsilon_{\pi,i} \in [0,1]^n$ that capture the (potential) error reduction from the projection operation in line 3.

Proof: 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, the total error per component is in the interval $[-n2^{-b}, 0]$. For the projection in line 3 we note that in view of Assumption 2 there is no arithmetic error; furthermore, every component of the previous error $\hat{t}_i - t_i$ can only be decreased (this can be verified analytically or graphically), giving rise to the LHS multiplication with diagonal matrix $\operatorname{diag}(\epsilon_{\pi,i})$. Finally, in line 4, the error 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). Putting everything together in terms of η_{i+1} and considering that $\hat{z}_0 - z_0 = \hat{y}_0 - y_0$ completes the proof. ∎

Although the errors η_i are inherently bounded by the box $\widehat{\mathbb{K}}$, we will see next that even if there is no truncation of the errors due to projection on $\widehat{\mathbb{K}}$, i.e. $\operatorname{diag}(\epsilon_{\pi,i}) = I$, $i = 0, 1, \ldots, I_{\max} - 1$, the errors remain bounded. In order to show this, we first express the two-step recurrence of the error in Proposition 2 as the one-step recurrence

$$\underbrace{\begin{bmatrix}\eta_{i+1}\\ \eta_i\end{bmatrix}}_{=:\xi_{i+1}} = \underbrace{\begin{bmatrix}(1+\hat{\beta})\big(I-\hat{H}_n\big) & -\hat{\beta}\big(I-\hat{H}_n\big)\\ I & 0\end{bmatrix}}_{=:A}\underbrace{\begin{bmatrix}\eta_i\\ \eta_{i-1}\end{bmatrix}}_{\xi_i} + \underbrace{\begin{bmatrix}(I-\hat{H}_n) & I\\ 0 & 0\end{bmatrix}}_{=:B}\underbrace{\begin{bmatrix}\epsilon_{y,i-1}\\ \epsilon_{t,i}\end{bmatrix}}_{=:\upsilon_i} \quad (8)$$

and prove that matrix A is Schur stable. Stability follows from the theorem below since matrix $\hat{H}_n$ is positive definite with maximum eigenvalue of less than or equal to one (Assumption 3) and step size $\hat{\beta}$ is in $[0, 1)$ (Assumption 5).

Theorem 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

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

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

Proof: Refer to [10]. ∎

The next lemma provides an upper bound on the magnitude of error η_i for *any* arithmetic round-off errors that might have occurred up to iteration i.

[2]Note that the approach in [21] assumes vanishing errors for an increasing iteration count; since we assume constant bit lengths for fixed-point number representation, this is not a viable approach in the context of this paper.

Lemma 1: Let b be the number of fraction bits and n be the dimension of the decision vector in problem (4). Consider the error dynamics due to arithmetic round-off in (8), 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

$$\bar{\eta}_i = \|EA^i\| \left\| \begin{bmatrix} \eta_0 \\ \eta_0 \end{bmatrix} \right\| + 2^{-b}\sqrt{n(1+n^2)} \sum_{k=0}^{i-1} \|EA^{i-1-k}B\| \quad (9)$$

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

Proof: From the one-step recurrence (8) 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 matrix A is Schur stable, the bound in (9) converges. Indeed, the effect of the initial error ξ_0 decays according to

$$\|EA^i\| \propto \rho(A)^i, \quad (10)$$

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

$$\sum_{k=0}^{i-1} \|EA^{i-1-k}B\| \propto \frac{1}{1-\rho(A)} - \frac{\rho(A)^i}{1-\rho(A)}. \quad (11)$$

This result can be used to construct the bound on the residuals in (7).

V. Parametrizable Hardware Architecture

Amdahl's law [7] states that the potential acceleration through parallelization is limited by the sequential fraction of an algorithm. In the FGM, a large fraction of the computation involves matrix-vector multiplications, hence the expected benefit of parallelization is substantial. In order to minimize sequential dependencies, the original FGM has been slightly modified (cf. Algorithm 1). Also observe that the computation of the individual vector components is independent of each other and the only communication occurs at matrix-vector multiplication. This allows for efficient parallelization given the custom computing and communication architecture discussed next. Specifically, we describe a tool that takes as inputs the data type, number of bits, level of parallelism and the latencies of an adder/subtracter (l_A) and multiplier (l_M) and automatically generates a digital architecture described in the VHDL hardware description language.

For a fixed-point data type, the generated architecture implementing Algorithm 1 for problem (4) is depicted in Figure 1. The matrix-vector multiplication is computed in the block labelled $\hat{v}^T\hat{w}$, which is shown in detail in Figure 2a. It consists of an array of n parallel multipliers followed by an adder reduction tree of depth $\lceil \log_2 n \rceil$. The architecture for performing the projection operation on set $\widehat{\mathbb{K}}$ is shown in Figure 2b. It compares the incoming value with the upper and lower bounds for that component. The results of the comparisons drive the select signal of a multiplexer which either saturates the component or allows it through.

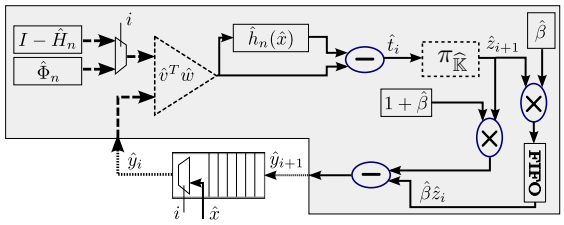
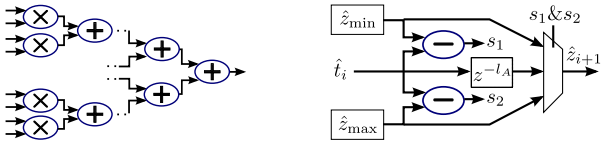


Fig. 1: Fast gradient compute architecture. Boxes denote storage elements and dotted lines represent n parallel vector links. The dot-product block $\hat{v}^T\hat{w}$ and the projection block $\pi_{\widehat{\mathbb{K}}}$ are depicted in Figure 2 in detail. FIFO stands for first-in first-out memory and 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 $\hat{h}_n(\hat{x})$ is stored in memory. In the subsequent iterations, the multiplexers allow $\hat{y}_i$ and $I - \hat{H}_n$ through and $\hat{h}_n(\hat{x})$ is read from memory.



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

(b) Projection block. A delay of l_A cycles is denoted by z^{-l_A}.

Fig. 2: Architectures of dot-product and projection block.

The amount of parallelism in the circuit is parameterized by parameter P. In Figure 1, $P = 1$, meaning that there is parallelism within each dot-product but the n 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, one computing the odd components of $\hat{y}_i$ and $\hat{z}_i$, the other computing the even. The different blocks communicate through a serial-to-parallel shift register that accepts P serial streams and outputs n parallel values for matrix-vector multiplication. These n values are the same for all blocks. It takes $\lceil \frac{n}{P} \rceil$ 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 $P \in \{1, \ldots, n\}$ is

$$L := \left\lceil \frac{n}{P} \right\rceil + l_A \lceil \log_2 n \rceil + 2l_M + 3l_A + 1. \quad (12)$$

Note that in a custom hardware implementation the number of execution cycles is exact. Providing cycle accurate completion guarantees is very important for high-speed real-time applications. Expression (12) also suggests that there will be diminishing returns to parallelization – a consequence of Amdahl's law. However, (12) 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.

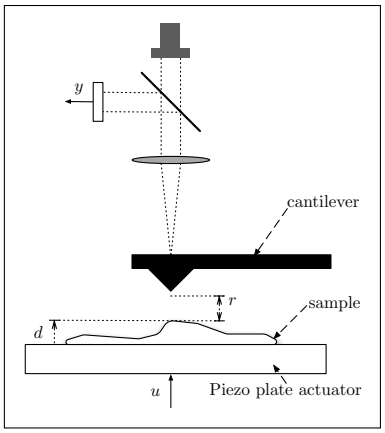


Fig. 3: Schematic diagram of the atomic force microscope (AFM) experiment. The signal u is the vertical displacement of the piezoelectric actuator, d is the sample height, r is the desired sample clearance, and y is the measured cantilever displacement.

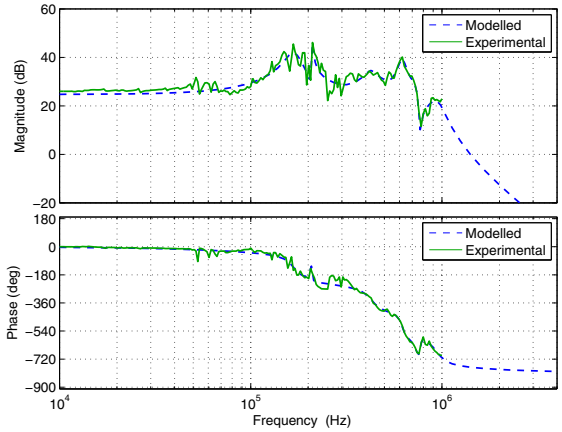


Fig. 4: Bode diagram for the AFM model (dashed, blue), and the frequency response data from which it was identified (solid, green).

VI. Case Study: Optimal Control of an Atomic Force Microscope

We consider the control of an AFM in which the overall objective is to obtain a topographical image of a sample specimen by measuring and manipulating the vertical clearance of a cantilever beam from the surface of the sample. The AFM system we consider is depicted schematically in Figure 3, in which the specific control objective during the imaging process is to maintain a constant reference distance $r = 50$ nm of the cantilever tip from the sample surface. The varying height d of the imaged sample can be controlled via the vertical displacement u of a piezoelectric plate actuator supporting the sample.

We use an experimentally obtained AFM system model from [12] whose frequency response is shown in Figure 4, along with the frequency response of a 12^{th} order LTI SISO model of the system. We use a state-space representation of this model in observer staircase form, so that the first

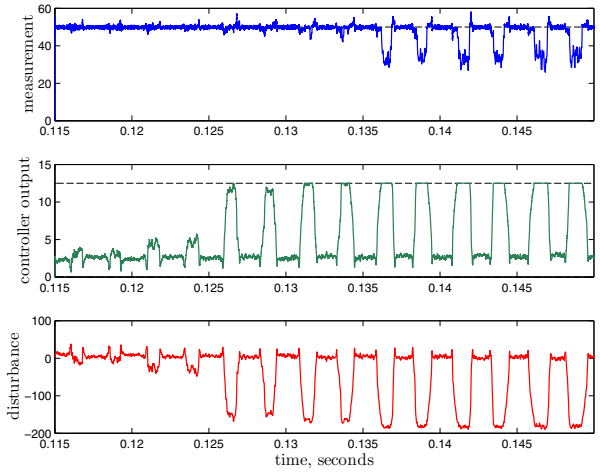


Fig. 5: Typical cantilever tip deflection (nm, top), control input signal (Volts, middle) and sample height variation (nm, bottom) profiles for the AFM example.

state is directly proportional to the controlled error signal $r-(d+y)$, in order to facilitate tuning of the controller via manipulation of the MPC objective function. We assume an input constraint $u \in [0,\ 12.5]$, representing the allowable input voltage range of the piezoelectric actuator. For the purposes of evaluating our FGM implementation of MPC, we assume that the system state is available from some external estimator. We choose a diagonal cost matrix Q and scalar R such that the system achieves the simulated closed-loop behavior exemplified by Figure 5 when the controller is implemented in a standard reference tracking configuration using a control horizon of 16 samples. To achieve good closed-loop performance, we target controller sampling rates in excess of 1 MHz.

Our goal is to choose the minimum number of bits and iterations such that the closed-loop performance is satisfactory while minimizing the amount of resources needed to achieve the desired sampling frequencies. Figure 6 shows the actual FGM convergence behavior of the residual $\|z^*(\hat{x}) - \hat{z}_i\|$ for one sample in the simulation. The maximum attainable accuracy for different numbers of bits is determined by the bound on the magnitude of any accumulation of round-off errors $\bar{\eta}_i$ in (9) since $\|z^*(\hat{x}) - \hat{z}_i\| \leq \|z^*(\hat{x}) - z_i\| + \bar{\eta}_i$. In the figure, we show the value of $\bar{\eta}_i$ at convergence, $\bar{\eta}_\infty$, as an indication of the tightness of our derived bound.

Table I shows the difference in closed-loop tracking performance for different fixed-point controllers compared to a double precision floating-point controller executing 400 fast gradient iterations at each sample (considered to achieve optimal tracking). It is clear that 15 iterations are not enough for satisfactory tracking. Assuming that a relative tracking error smaller than 0.1% is desirable, using 20 fast gradient iterations and 14 fraction bits would be the optimal choice.

For a fixed number of iterations one can calculate the execution time deterministically according to (12). The FPGA designs can be clocked at more than 400 MHz using chips from Xilinx's high-performance Virtex 6 family or at more

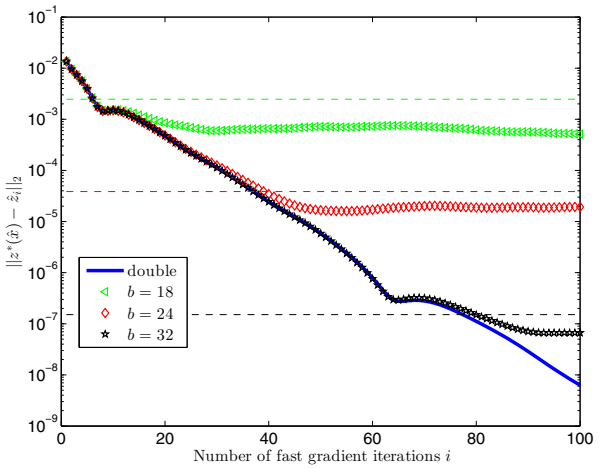


Fig. 6: Convergence of the fast gradient method and round-off error bound $\bar{\eta}_\infty$ given by (9) for different bit lengths.

TABLE I: Percentage difference between the tracking error for a double precision floating-point controller using $I_{\max} = 400$ and different fixed-point controllers.

$I_{\max} \backslash b$	10	12	14	16	18	20	22
15	55.18	33.25	29.13	28.74	29.28	29.25	30.65
20	16.13	0.88	0.06	0.02	0.02	0.02	0.02
25	17.56	0.96	0.05	0.01	0.01	0.01	0.01
30	17.57	0.96	0.04	0.00	0.00	0.00	0.00
35	17.42	0.95	0.04	0.00	0.00	0.00	0.00

than 230 MHz using the low cost and low power Spartan 6 family. Table II shows the achievable sampling times 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. With Virtex 6 devices one can achieve sampling times beyond 1 MHz for $P = 2$ and close to 2 MHz for $P = 16$ (maximum parallelism), whereas for Spartan 6 devices well over 600 kHz sampling frequencies are achievable with $P = 2$ and close to 1 MHz for $P = 7$. For Virtex 6, all designs fit inside the smallest device in the family (LX75T), whereas for Spartan 6 technology a variety of chips will be suitable for different designs. Note that the devices in the low power family will have power ratings in the region of 1 Watt.

TABLE II: Resource usage and potential performance at 400MHz (Virtex 6) and 230MHz (Spartan 6) with $I_{\max} = 20$. T_s stands for sampling time.

P	1	2	3	4	6	7	16
multipliers	18	36	54	72	108	126	288
V6 T_s (μs)	1.30	0.90	0.80	0.70	0.65	0.60	0.55
S6 T_s (μs)	2.26	1.57	1.39	1.21	1.13	1.04	-
S6 chip	LX16	LX25	LX45	LX75	LX75	LX75	-

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. The authors wish to thank Abu Sebastian of IBM Zürich and Stefan Kuiper for experimental data and technical advice related to the AFM example.

REFERENCES

[1] LogiCORE IP floating-point operator v5.0, 2011.
[2] M. Baes. Estimate sequence methods: Extensions and approximations, Nov. 2009.
[3] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, Jan 2002.
[4] O. Devolder, F. Glineur, and Y. Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Submitted to Mathematical Programming*, 2011.
[5] 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.
[6] 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.
[7] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 5th edition, 2011.
[8] 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.
[9] 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.
[10] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari. Embedded online optimization for model predictive control at megahertz rates. *arXiv:1303.1090*, Mar 2013.
[11] M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In *Proc. 18th IFAC World Congress*, Milano, Italy, Aug 2011.
[12] S. Kuiper. *Mechatronics and Control Solutions for Increasing the Imaging Speed in Atomic Force Microscopy*. PhD thesis, Delft University of Technology, 2012.
[13] J. M. Maciejowski. *Predictive Control with Constraints*. Pearson Education, Harlow, UK, 2001.
[14] J. Mattingley, Y. Wang, and S. Boyd. Receding horizon control: Automatic generation of high-speed solvers. *IEEE Control Systems Magazine*, 3(31):52–65, 2011.
[15] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.
[16] Y. Nesterov. A method for solving a convex programming problem with convergence rate $1/k^2$. *Soviet Math. Dokl.*, 27(2):372–376, 1983.
[17] Y. Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course*. Springer, 2004.
[18] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *Proc. 51st IEEE Conf. on Decision and Control*, Maui, HI, USA, Dec 2012.
[19] S. Richter, C. Jones, and M. Morari. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Trans. Aut. Control*, 57(6), Jun 2012.
[20] 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*, pages 5223–5229, Orlando, USA, Dec 2011.
[21] M. Schmidt, N. L. Roux, and F. Bach. Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization. *arXiv:1109.2415v2*, Dec 2011.
[22] 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 Trans. Control Sys. Tech.*, 17(5):1006–1017, Sep 2009.
[23] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Number 32 in Notes on Applied Science. Her Majesty's Stationary Office, London, UK, 1st edition, 1963.
[24] A. G. Wills, G. Knagge, and B. Ninness. Fast linear model predictive control via custom integrated circuit architecture. *IEEE Trans. Control Sys. Tech.*, 20(1):59–71, 2012.