Fast algorithms for constrained generalised predictive control with on-line optimisation

Vinicius Berndsen Peccin1 | Daniel Martins Lima2 | Rodolfo César Costa Flesch1

1 Department of Automation and Systems, Federal University of Santa Catarina, Florianópolis, Santa Catarina, Brazil
2 Department of Control, Automation and Computing, Federal University of Santa Catarina, Blumenau, Santa Catarina, Brazil

Correspondence
Julio Elias Normey-Rico, Department of Automation and Systems, Federal University of Santa Catarina, Florianópolis, Santa Catarina, Brazil.
Email: julio.normey@ufsc.br

Funding information
Conselho Nacional de Desenvolvimento Científico e Tecnológicos, Grant/Award Numbers: 304032/2019-0, 305785/2015-0, 309244/2018-8; Chinese National Natural Science Foundation, Grant/Award Number: 62071124

Abstract
This paper proposes fast algorithms for constrained generalised predictive control based on two first-order methods, namely accelerated dual gradient projection method and fast alternating minimisation algorithm. Theoretical bounds on the number of iterations, which play an important role in the context of real-time model predictive controllers, are provided for both algorithms. Also, some implementation issues in parallel architectures are discussed. The methods are firstly validated by simulation and their results are compared with the ones presented by commercial solvers. A three-phase grid-connected LCL-filtered inverter was used as a case study. The algorithms were evaluated in an FPGA with the quadratic program computed in microseconds.

1 | INTRODUCTION

Nowadays, model predictive control (MPC) is still the most relevant advanced control method in industry [1]. However, MPC is mostly used to control slow dynamics processes with large sampling times and sufficient computational resources [2]. This can be explained, in part, because the control signal computation of MPC, in its conventional formulation, requires a quadratic programming optimisation to be solved. This issue motivated researches to find alternatives to accelerate the computation time of MPC algorithms and implementations. More recently, with the increase in the availability of high-performance computational systems, the application of MPC algorithms in processes with fast dynamics is becoming a reality.

Some researches on fast constrained MPC exploit the quadratic program (QP) particularities to compute it very efficiently. One approach is to solve a set of QP offline and store the optimal values in a look-up table, such as explicit MPC in [3]. However, this kind of solution is limited to small problems due to memory limitations [4]. Another approach is to exploit the structure of the resulting problem. Among on-line QP solvers, the first- and second-order methods deserve to be highlighted. Some works propose second-order methods for fast MPC, such as the ones based on active-set (see [5, 6]) and interior-point methods (see [2]). However, second-order methods require complex linear algebraic routines that result in complex algorithms, which make use of a large amount of code memory [7]. On the other hand, first-order methods are cheaper on the computational requirements and have become a good choice for embedded fast MPC. Also, for large problems, they can be used in distributed MPC applications [8]. The most widely used first-order algorithms for MPC applications are the operator splitting method [9] and the gradient projection method [10]. In general, first-order methods must be used when a solution of low to medium accuracy is sufficient [11]. However, their use in practical closed-loop control systems is suitable due to the feedback action, and physical limitations such as the resolution of the sensors and the analog to digital converter. Another drawback of first-order methods is that accelerated versions, based on Nesterov’s method [12], can become unstable because of the error...
accumulation if fixed-point arithmetic is used. To overcome this limitation, the number of integer and fractional bits must be carefully chosen. Some guidelines to choose the fixed-point word size, for a dual gradient method, are presented in [13].

The research in fast MPC is mostly focused on state-space formulations. However, generalised predictive control (GPC) is one of the most used MPC formulations in industry and makes use of discrete-time transfer functions to calculate the predictions [14]. Transfer-function formulations are widespread in industry, since concepts such as dead time, gains and time constants are more familiar in industrial context than the ones associated with state-space formulations [14]. Another important issue is that dead time, common in industrial processes, can be easily represented in the transfer function form, which is not true for other approaches [15]. Some recent works for fast GPC applied to frequency inverters are presented in [16] and [17], however only the unconstrained case was treated. Some examples of MPC used for three-phase power converter control are presented in [18], where a robust unconstrained MPC with resonant controllers are used to achieve harmonic rejection, and in [19], where a reduced model unconstrained MPC with an embedded integrator and a feed-forward term is introduced to minimise the harmonic content of the grid currents.

The main objective of this paper is to propose a GPC algorithm integrated with first-order methods to solve the resulting QP at each sampling instant. The first-order methods used were the accelerated dual gradient projection (GPAD) method and the fast alternating minimisation algorithm (FAMA). The resulting algorithms are called GPCGPAD and GPCFAMA, and they can be used to control systems with fast dynamics. In addition, they are tailored for embedded applications with characteristics such as simplicity to generate a software able to be verified, validated and certified, limitation on the memory space to keep data that define the code which implements the solution; and determinism with a predictable worst case execution time (WCET), in order to meet hard real time requirements. An advantage of the algorithms in comparison with state-space based solutions with first-order methods, such as [7–10], is the smaller number of sensors and variables required by the prediction model used by GPC. GPCGPAD was previously presented in [20], and in this work its results is extended for multi-input multi-output (MIMO) processes. As the formulation used in the proposed approach is based on transfer function models, the extension to the MIMO case is not as straightforward as it is in the case of state-space models. In addition, the generalisation proposed in this paper allows considering a different prediction horizon for each process variable and a different control horizon for each manipulated variable. Also, analysis of theoretical bounds on the number of iterations, some details of implementation, and a preconditioning method are discussed. Moreover, a comparison with GPCFAMA is presented, which is also a novel contribution of this work. The five main contributions of this work are:

- the proposition of efficient constrained GPC algorithms with GPAD and FAMA, tailored for implementation using fast speed hardware with on-line optimisation;
- a comparison of the proposed strategy with other MPC approaches and general-purpose solvers;
- deterministic analysis of WCET for the proposed implementations in an FPGA based on the theoretical bound on the number of iterations for the algorithms given in the literature;
- validation with a three-phase grid-tie LCL-filtered inverter case study, which is a system with fast dynamics that plays a fundamental role in renewable energy applications;
- discussion of implementation features such as parallel matrix multiplication and a preconditioning approach that allow to reduce the computation time for the FPGA architecture considered in this study.

The rest of the paper is organised as follows: Section 2 presents the proposed GPC formulation with the accelerated dual gradient projection method, and with the fast alternating minimisation algorithm. Also, the algorithms for their implementations are presented. Section 3 describes some implementation details of the proposed algorithms. Section 4 presents a case study modelled in Simulink/MATLAB® and simulated with the proposed algorithms in order to validate and compare them with commercial solvers. In Section 5, an implementation in a field programmable gate array (FPGA) is presented and computation time requirements are discussed. Section 6 concludes the paper.

Notation: let \( \mathbb{R}, \mathbb{R}^+, \mathbb{R}^\infty, \mathbb{R}^{\infty\times m} \) denote the sets of real numbers, the set of negative real numbers, the column real vectors of length \( n \) and, the real matrices of length \( n \times m \), respectively. For a vector \( \mathbf{v} \in \mathbb{R}^n \), \( v_i \) denotes the \( i \)th element, and for a matrix \( \mathbf{M} \in \mathbb{R}^{\infty\times m} \), \( M_{ij} \) denotes the element at the \( i \)th row and \( j \)th column. An identity matrix of length \( n \times m \) is denoted by \( I_{\infty\times m} \). A vector where all elements are zero is denoted by \( \mathbf{0} \), with appropriate dimensions. The transpose of a matrix \( \mathbf{M} \) is denoted by \( \mathbf{M}^T \). For a positive definite matrix \( \mathbf{M} \), \( \lambda_{\text{max}}(\mathbf{M}) \) and \( \lambda_{\text{min}}(\mathbf{M}) \) denote the largest and smallest eigenvalue of \( \mathbf{M} \), respectively. For vectors, the operators \( \max \) and \( \min \) are defined at element-wise. The Euclidean norm of a vector \( \mathbf{v} \) is denoted by \( \| \mathbf{v} \| \), while the spectral norm of a matrix \( \mathbf{M} \) is denoted by \( \| \mathbf{M} \| \).

For a sampled signal \( x \), \( x(k) \) is the value at the \( k \)th sampling instant. For a variable \( v \), \( v^{(i)} \) denotes the \( i \)th instance of the same variable. For algorithm purposes, the symbol \( : \) denotes a value assignment to a variable.

## 2 Proposed Algorithms

A MIMO system with \( n_u \) inputs and \( n_y \) outputs can be represented by the controlled autoregressive integrated moving average (CARIMA) model as follows:

\[
\mathbf{A}(z^{-1}) \mathbf{y}(k) = \mathbf{D}(z^{-1}) \mathbf{B}(z^{-1}) u(k-1) + \mathbf{C}(z^{-1}) \frac{e(k)}{\Delta},
\]

where \( \mathbf{y} \in \mathbb{R}^{n_y} \) is a vector of the system outputs, \( \mathbf{u} \in \mathbb{R}^{n_u} \) is a vector of the system inputs, \( \mathbf{A}(z^{-1}) \) is a diagonal polynomial
matrix of dimensions \( n_e \times n_e \), polynomial matrix \( B(\zeta^{-1}) \) has dimensions \( n_e \times n_y \), and polynomial matrix \( C(\zeta^{-1}) \) has dimensions \( n_y \times n_e \). \( D(\zeta^{-1}) \) is a diagonal matrix of dimensions \( n_e \times n_e \), with each element representing the smallest dead time between the inputs related to the corresponding output, \( \Delta = 1 - \zeta^{-1} \) is used to improve the modelling of non-stationary disturbances.

The typical cost function of a MIMO GPC is given by:

\[
J_{GPE} = \sum_{p=1}^{n_p} \sum_{k=1}^{N_2} \left[ \tilde{y}_p(k + b|k) - w_p(k + b) \right]^2 q_{\delta}^{(p)} + \sum_{i=1}^{n_i} \sum_{b=1}^{N_l} \left[ \Delta u_l(k + b - 1) \right]^2 q_{\lambda}^{(i)},
\]

where \( \tilde{y}_p(k + b|k) \) represents the prediction of the system output \( p \) at a future discrete-time instant \( k + b \), determined at \( k \), \( w_p(k + b) \) is the reference trajectory at \( k + b \), \( \Delta u_l(k) \) is the control increment for input \( l \), \( N_2 = N_2^{(p)} - N_1^{(p)} + 1 \) defines the prediction horizon for output \( p \), where \( N_1^{(p)} \) is the initial horizon sample and \( N_2^{(p)} \) is the final horizon sample. \( N_l^{(i)} \) is the control horizon for input \( l \). Assuming constant weights over the horizons, \( q_{\delta}^{(p)} \) and \( q_{\lambda}^{(i)} \) are error and control effort weights, which can be grouped in positive definite diagonal block matrices \( Q_{\delta}^{(p)} \) and \( Q_{\lambda}^{(i)} \), respectively.

If we define the control increment generalised vector as:

\[
\Delta u = [\Delta u^{(1)} \Delta u^{(2)} \ldots \Delta u^{(n_i)}]^T, \quad N_u = \sum_{i=1}^{n_i} N_u^{(i)},
\]

the typical GPC QP can be represented by:

\[
\begin{align*}
\min_{\Delta u} & \quad f(\Delta u) = \frac{1}{2} \Delta u^T H \Delta u + b^T \Delta u \\
\text{s.t.} & \quad R \Delta u \leq r,
\end{align*}
\]

where \( R \in \mathbb{R}^{n_u \times N_u} \) is the constraints matrix, \( r \in \mathbb{R}^{n_r} \) is the constraints vector, and \( n_{io} \) is the number of inequality constraints. The condensed form to represent the constraints \( R \Delta u \leq r \) is generic and allows the representation of many types of constraints, such as magnitude, slew rate, and overshoot of both input and output variables [14]. The details to obtain (2) and the GPC formulation for MIMO systems are presented in [21] and [22].

The set of QPs from GPC defined in (2), considering all control iterations, can be reformulated as a parametric optimisation of only one parameter with a proper change of variables. This representation is used to obtain the sub-optimality bounds on the number of iterations for the proposed algorithms. Considering a constant set-point \( w \), the GPC QP, for all control iterations, can be viewed as a parametric optimisation problem:

\[
F^*(\epsilon) = \min_{\Delta u} \frac{1}{2} \Delta u^T H \Delta u + 2(\epsilon e - w)^T Q_{\delta} G \Delta u
\]

\[
\text{s.t.} \quad R \Delta u \leq \bar{r} + \tilde{r},
\]

where the feedback variable \( e \), as a way to represent the state of the system, is composed of the present output and finite series of past inputs and outputs and is given by \( e = [y(k) \cdots y(k - n_y)]^T \). Comparing with (2), the vectors \( b \) and \( r \), which are functions of \( e \), now are represented as \( b^T = 2(\epsilon e - w)^T Q_{\delta} G \) and \( r = \bar{r} + \tilde{r} e \). The matrix \( E \) can be obtained from the matrix of coefficients from the polynomial matrices \( F(\zeta^{-1}) \) and \( S(\zeta^{-1}) \) as \( E = [F \ S] \). \( F(\zeta^{-1}) \) and \( S(\zeta^{-1}) \) can be obtained from a solution of a Diophantine equation as explained in [14, p. 49]. The matrix \( F \) is used when there are constraints which depend on the state of the system, such as output or control signal constraints. If only constraints on control increments are considered, then \( F \) is set as a matrix of zeros. It is important to note that the matrices and vectors which define the constraints, costs and dynamics in (3) do not vary during the execution of the GPC controller and can be precomputed off-line.

### 2.1 GPCGPAD algorithm

The gradient projection method (GPM) is similar to the descent gradient method and can be used to solve constrained convex optimisation problems. The GPAD method presented in [10], applies the accelerated dual form on the gradient projection method.

Considering the GPC optimisation problem (2), the first step to obtain the dual form is to define the associated Lagrangian:

\[
\mathcal{L}(\Delta u, \psi) = \frac{1}{2} \Delta u^T H \Delta u + b^T \Delta u + \psi^T (R \Delta u - r),
\]

where \( \psi \in \mathbb{R}^r \) is the Lagrange multipliers associated to the inequality constraints.

The dual optimisation problem, presented as a minimisation, can be stated as:

\[
\begin{align*}
\min_{\psi} & \quad -f_d(\psi) = \frac{1}{2} \psi^T (H^{-1} R^T \psi + (R^{-1} b + r)^T \psi \\
\text{s.t.} & \quad \psi \geq 0,
\end{align*}
\]

which is a QP with strong duality, from the Slater’s condition, since there is a strictly feasible point in the primal problem [23, Sec. 5.2.3].
The main step of the GPM applied to (5) becomes:

$$\psi_{j+1} = P\left(\psi_j - \frac{1}{L} \nabla f_d(\psi_j)\right),$$  \hspace{1cm} (6)

where $j$ is an iteration index, $P$ is an orthogonal projection and $L \geq 0$ is a Lipschitz constant, which ensures convergence with a feasible step in descent direction [10]. An approach to obtain $L$ is the Frobenius norm of the Hessian:

$$L = \sqrt{\sum_{i,j=1}^{n} |\nabla^2 f(\mathbf{x})_{ij}|^2},$$  \hspace{1cm} (7)

where $n$ is the size of the square matrix and $i, j$ are the indices of the Hessian elements.

Since the constraints associated with (5) are only $\psi \geq 0$, the projection is equivalent to a saturation of $\psi$ to ensure positive values [10], which can be represented mathematically as:

$$P(\psi) = \max(\psi, 0).$$  \hspace{1cm} (8)

Thus, (6) can be rewritten as:

$$\psi_{j+1} = \max\left(\psi_j - \frac{1}{L} \left(RH^{-1} R^T \psi_j + RH^{-1} b + r\right), 0\right),$$  \hspace{1cm} (9)

which can be divided in two steps:

$$\Delta u_j = -H^{-1}(b + R^T \psi_j),$$

$$\psi_{j+1} = \max\left(\psi_j + \frac{1}{L} (R\Delta u_j + r), 0\right).$$  \hspace{1cm} (10)

The acceleration step was proposed by Nesterov [12] and has been widely used in the context of fast MPC. The main point for the acceleration of the method is the non-observance of the relaxation sequence property and its replacement by the so-called estimated sequence. In this way, an *a priori* operation is inserted in the calculation:

$$\mathbf{w} = \psi_j + \beta (\psi_j - \psi_{j-1}),$$  \hspace{1cm} (11)

where $\beta$ is defined as $\beta = \frac{j-1}{j+2}$.

For the stop criteria of the algorithm, primal feasibility and primal optimality can be used. For primal feasibility it is possible to test if the constraints are met within constraints tolerance $\epsilon_f$, that is:

$$R\Delta u_j - r \leq \epsilon_f.$$  \hspace{1cm} (12)

For the primal optimality criterion, a property derived from the strong duality, can be used, which implies:

$$f(\Delta u) - f(\Delta u^*) \leq f(\Delta u) - f_d(\psi).$$  \hspace{1cm} (13)

A solution is considered $\epsilon_o$-optimal if:

$$f(\Delta u) - f_d(\psi) \leq \epsilon_o,$$  \hspace{1cm} (14)

so by substituting $f_d(\psi)$ from definition (5) and the cost function $f(\Delta u)$ from (2) in (14) the solution will be $\epsilon_o$-optimal if:

$$-\psi^T (R\Delta u_j - r) \leq \epsilon_o.$$  \hspace{1cm} (15)

From the presented developments and the recursive algorithm to obtain the GPC matrices detailed in [21], GPCGPAD can be summarised as in Algorithm 1, where $j_{\text{max}}$ is the maximum number of iterations and $\tilde{u}_i$ is a unit step signal with $\tilde{u}_i = 1$ for $i \geq 0$ and $\tilde{u}_i = 0$ for $i < 0$.

The sub-optimality bounds on the number of iterations for GPAD were proven in [10]. If they are applied to the problem in (3), considering $\psi_0 = 0$, the upper bound for the number of iterations can be written as:

$$N_j(\epsilon, \epsilon_o) = \left[\frac{8L\left\|\psi^*(\epsilon)\right\|_1}{\epsilon_f}\right] - 2,$$  \hspace{1cm} (16)

that guarantees the averaged primal solution $\Delta u_j$ to be feasible within a prescribed threshold $\epsilon_f$. Another upper bound is

$$N_j(\epsilon, \epsilon_o) = \left[\frac{2L\left\|\psi^*(\epsilon)\right\|_1}{\epsilon_o}\right] - 2,$$  \hspace{1cm} (17)

which guarantees the averaged primal solution $\Delta u_j$ to be $\epsilon_o$-optimal.

From (16) and (17) one can define an upper iteration bound $N_j(\epsilon, \epsilon_o)$

$$N_j(\epsilon, \epsilon_o) \geq \sup_{\epsilon \in \mathcal{E}} \{N_j(\epsilon, \epsilon_f), N_j(\epsilon, \epsilon_o)\}.$$  \hspace{1cm} (18)

It is important to notice that, given a GPC problem, all quantities to obtain the upper iteration bound are known except for $\psi^*(\epsilon)$. The upper iteration bound for a given polyhedron of past conditions, $\mathcal{E}$, depends on the value of $\epsilon$ which produces the largest optimal dual solution $\psi^*(\epsilon)$. In order to find an upper bound for the dual solution $\Delta \psi(\mathcal{E})$, where:

$$\left\|\psi^*(\epsilon)\right\|_1 \leq \Delta \psi(\mathcal{E}),$$  \hspace{1cm} (19)

it is possible to solve [24]:

$$\Delta \psi(\mathcal{E}) = \max_{\psi, \epsilon} \sum_{j=1}^{j_{\text{max}}} \psi_j,$$

s.t. $\epsilon \in \mathcal{E}$,

$$\psi = \max(\psi_k - \frac{1}{L} \nabla f_d(\psi_k), 0).$$  \hspace{1cm} (20)
Algorithm 1 Generalised predictive control with GPAD

Input: $A^{(1)}, \ldots, A^{(n_o)}, B^{(1)}, \ldots, B^{(n_o+n_x)}, D, R, r$
Output: $u^{(1)}(k), \ldots, u^{(n_o)}(k)$
Data: $n_o, n_c, Q_0, Q_f, N_1, n_1, N_2, n_2, z^0 = 0, f, e, n_{\text{max}}$

begin
  for $p = 1 : n_o$ do
    $A^{(p)}(z^{-1}) := (1 - z^{-1})A^{(p)}(z^{-1})$;
  end
  for $l = 1 : n_c$ do
    for $i = 1 : N_2$ do
      $g^{(p)}(i) := z \left( 1 - A^{(p)}(z^{-1}) \right) g^{(p-1)}(i)$ + $B^{(p-1)}(z^{-1}) v_{i}$;
    end
  end
  for $i = 1 : n_1$ do
    $G_{(i)} := \left[ g^{(p)}(i), N^{(i)}_{N_1}, N^{(i)}_{N_1} + 1, \ldots, N^{(i)}_{N_1} + n_2 \right] ;$
    $H := 2 (GT_{Q_1}G + Q_1); H := H^{-1};$
    $L := \sqrt{\sum_{i,j=1}^{n_1,n_2} RHR^T / 2};$
    $k := 0;$
  while automatic mode do
    if sample time then
      for $p = 1 : n_o$ do
        Acquire outputs $y^{(p)}(k)$ and ref. $w^{(p)}(k)$;
        $f^{(p)} := y^{(p)}(k);$;
        for $i = 1 : N_2$ do
          $f^{(p)} := \left( 1 - A^{(p)}(z^{-1}) \right) f^{(p-1)}(i)$ + $B^{(p-1)}(z^{-1}) \Delta u^{(p)}(k - D^{(p)} + 1);$;
        end
        $f := \left[ f^{(1)}(1) \ldots f^{(n_o)}(n_2) \right] ;$
        $b^T = 2 (f - w)^T Q_1 G;$
        while $j < n_{\text{max}}$ do
          if $j = 0$ then
            $\beta := 0$
          else
            $\beta := \frac{1}{j + 1}$
          end
          $\Delta u := -H (R^T w + b^T);$;
          $s := (1 / L) (R \Delta u - r);$;
          if $s_i \leq \epsilon_f / L, \forall i = 1 \ldots n_{\text{max}}$ then
            if $-w^T s < \epsilon_o / L$ then
              return;
            end
            $\psi_{j+1} := \max (w + s, 0);$;
          end
          $j := j + 1;$
          for $l = 1 : n_c$ do
            $\Delta u^{(l)}(k) := \Delta u_l;$
            $i := i + N_{u}^{(l)};$
            $u^{(l)}(k) := u^{(l)}(k - 1) + \Delta u^{(l)}(k);$;
            $k := k + 1;$
          end
        end
      end
    end
  end
end

As shown in [24], (20) can be reformulated as a mixed integer linear programming (MILP) by introducing a binary vector such that $[\delta_{i} = 1] \leftrightarrow [\psi_{k} := \frac{1}{L} \int_{k}^{\infty} f_d(\psi_{k}) > 0]$. The resulting MILP to obtain the upper bound for $\| \psi^{*}(e) \|$ of the proposed algorithm is given by:

$$\min_{\psi, e} \sum_{i=1}^{n_o} \psi_i$$

s.t. $\psi_i \geq 0,$

$$\psi_i \leq \lambda_i^* \delta_i,$$

$$\frac{1}{L} \int_{k}^{\infty} f_d(\psi_{k}) \leq -\frac{1}{L} (be + r) + \mu_i^* (1 - \delta_i),$$

$$\frac{1}{L} \int_{k}^{\infty} f_d(\psi_{k}) \geq -\frac{1}{L} (be + r) + \mu_i^* (1 - \delta_i).$$

2.2 GPCFAMA algorithm

The Fast Alternating Minimisation Algorithm (FAMA) was proposed in [25] and was developed to solve optimisation problems as follows:

$$\min_{x, z} h(x) + g(z)$$

s.t. $C x + D z = c$.

(22)

Function $h$ is smooth convex and function $g$ convex. The acceleration is obtained by over-relaxing the Lagrange multiplier vectors after each iteration. FAMA can be viewed as an accelerated scheme equivalent to performing fast iterative shrinkage-thresholding algorithm (FISTA) on the dual [25]. Thus, by applying the Nesterov-type acceleration step to the dual variable $\theta$, FAMA is obtained and its steps can be summarised as:

$$\mathbf{x}_{j+1} := \text{argmin} \mathcal{L}(\mathbf{x}_j, \mathbf{z}_j, \hat{\theta}_j),$$

$$\mathbf{z}_{j+1} := \text{argmin} \mathcal{L}_\rho(\mathbf{x}_{j+1}, \mathbf{z}_j, \hat{\theta}_j),$$

$$\mathbf{v}_{j+1} := \mathbf{v}_j + \rho (\hat{\mathbf{C}} \mathbf{x}_{j+1} + \hat{\mathbf{D}} \mathbf{z}_{j+1} - \hat{e}),$$

$$\hat{\theta}_{k+1} := \left( 1 + \frac{\sqrt{4(\theta_k)^2 + 1}}{2} \right) / 2,$$

$$\hat{\psi}_{j+1} := \mathbf{v}_{j+1} + (\hat{\theta} - 1) (\mathbf{v}_{j+1} - \mathbf{v}_j) / \hat{\theta}_{k+1}.$$  

(23)

In order to write the optimisation problem (2) in the form of (22) so that the results of FAMA can be used, the first step proposed is to represent the inequality constraints using an indicator function $I_\_$. Defining $f(\Delta u) = \frac{1}{2} \Delta u^T H \Delta u + b^T \Delta u$, (2) can be rewritten as:

$$\min_{\Delta u} f(\Delta u) + I_\_ (R \Delta u - r).$$

(24)

A new variable $\mathbf{z} = R \Delta u - r$ and a new equality constraint can be defined to ensure the coupling between the variables:

$$\min_{\Delta u, \mathbf{z}} f(\Delta u) + I_\_ (\mathbf{z})$$

s.t. $R \Delta u - \mathbf{z} = \mathbf{r}.$

(25)

From the standard problem (25), the Lagrangian $\mathcal{L}$ and the augmented Lagrangian $\mathcal{L}_\rho$ can be defined as:
\[ L_{\rho}(\Delta u, z, v) = f(\Delta u) + I_{-}(z) + v^T (R\Delta u - z - r), \]

\[ \Delta u_{j+1} := \text{argmin} \left( \Delta u_j^T H \Delta u_j + (b^T + \hat{y}_j^T R) \Delta u_j \right), \]

\[ z_{j+1} := \text{argmin} \left( I_{-}(z) + \hat{y}_j^T z_j + \frac{\rho}{2} \| R \Delta u_{j+1} - z_j - r \|^2 \right), \]

\[ \psi_{j+1} := \psi_j + \rho (R \Delta u - z - r), \]

\[ \theta_{k+1} := (1 + \sqrt{4(\theta_k)^2 + 1})/2, \]

\[ \hat{y}_{j+1} := \psi_{j+1} + (\theta_{k+1} - 1)(\psi_{j+1} - \psi_j)/\theta_{k+1}. \]

The step in (26) can be obtained analytically in a direct way:

\[ \Delta u_{j+1} := -H^{-1}(b + R^T \hat{y}_j). \]

In a typical GPC problem, the Hessian \( H \) can be computed off-line and its inverse can be stored in memory. So, this step is obtained with simple matrix vector multiplication.

To compute step (27), notice that this kind of problem can be rewritten as a proximal operator [26]. The proximal operator of a scaled function \( \rho h(x) \) is defined as:

\[ \text{prox}_{\rho h}(x) = \text{argmin}_z \left( h(z) + \frac{1}{2\rho} \| z - x \|^2 \right). \]

So, applying the proximal operator to an indication function \( I_{-} \) and a step size \( \rho \), step (27) can be represented as

\[ z_{j+1} := \text{prox}_{\rho I_{-}}(R \Delta u_{j+1} - r + (1/\rho) \hat{y}_j). \]

Given that \( I_{-}(z) \) is an indicator function of the negative orthant \( \mathbb{R}^n_- \), which is a closed nonempty convex set, the proximity operator becomes a projection onto \( \mathbb{R}^n_- \) [27]. So, (32) can be obtained by taking the negative part of each component or zero:

\[ z_{j+1} := \min (R \Delta u_{j+1} - r + (1/\rho) \hat{y}_j, 0). \]

The residuals of the optimality conditions can be used as stopping criteria. The primal residual is given by:

\[ r_{j+1} := R \Delta u_{j+1} - z_{j+1} - r, \]

and the dual residual is:

\[ s_{j+1} := \rho R^T (z_{j+1} - z_j). \]

So, a reasonable termination criterion is that the primal and dual residuals must be smaller than desired tolerances:

\[ \| r_j \| \leq \varepsilon_r \quad \text{and} \quad \| s_j \| \leq \varepsilon_d. \]

To ensure convergence in the dual objective with rate \( O(1/k^2) \), the parameter \( \rho \) must be \( \rho < \sigma_f/H \lambda_{max}(R^T R) \) with \( \sigma_f \) being the strong convexity constant of \( H \) and \( \lambda_{max} \) the largest eigenvalue of the matrix [25]. Assuming \( H \) strongly convex, \( \rho \) can be obtained as \( \rho = \lambda_{min}(H)/\lambda_{max}(R^T R) \).

From the presented developments and the recursive algorithm to obtain the GPC matrices detailed in [21], GPCFAMA can be summarised as in Algorithm 2.

The convergence rate for FAMA was proven in [25] and is \( O(1/k^2) \) for the dual cost \( f_d(\hat{\psi}^*) - f_d(\hat{\psi}) \). From Theorem 5 in [25] the sub-optimality bound for FAMA which provides the upper bound on the dual cost \( N_d(e, \varepsilon_d) \), for \( \psi_0 = 0 \), is

\[ N_d(e, \varepsilon_d) = \sqrt{\frac{2\lambda_{max}(R^T R)}{\lambda_{min}(H)e^2} \| v^*(e) \|} - 1, \]

which guarantees the solution \( \psi_j \) to be optimal within a prescribed threshold \( \varepsilon_d \).

In Theorem 5.3 from [28] the convergence rate for the squared Euclidean distance on the primal variable \( \| \Delta u_j - \Delta u^* \|^2 \) was presented with \( O(1/k^2) \). Thus, another upper bound on the number of iterations that can be used is \( N_p(e, \varepsilon_p) \), defined as:

\[ N_p(e, \varepsilon_p) = \left( \frac{2\lambda_{max}(R^T R)}{\lambda_{min}(H)e^2} \| v^*(e) \|, \right), \]

which guarantees the squared Euclidean distance on the primal variable to be optimal within a prescribed threshold \( \varepsilon_p \).

The upper iteration bound depends on \( e \) and the optimal dual solution \( v^*(e) \). In order to find an upper bound for the worst case of \( v^*(e) \) with \( e \) in a given polyhedron \( \mathcal{E} \), two approaches are proposed in [28]. The first one uses sum of squares relaxations and the second one uses sample-based estimation. The sample-based estimation approach can be easily applied for all problem dimensions. The procedure is to choose a confidence parameter \( \tilde{\beta} \in [0, 1] \), a level parameter \( e \in [0, 1] \) and take \( N_j \) satisfying \( N_j \geq \frac{1}{e^2} - 1 \). The next step is randomly drawing \( N_j \) samples \( \{e_1, ..., e_N\} \in \mathcal{E} \) and run the Algorithm 2 to calculate the corresponding \( \{\| v^*(e_1) \|, ..., \| v^*(e_N) \| \} \). Finally, one can compute \( \max_{1 \leq k \leq N} \{\| v^*(e_k) \| \} \) which is an upper-bound of an optimal solution with \( e \)-level robust feasibility for \( v^*(e) \).
3 IMPLEMENTATION ISSUES

This section describes some implementation details of the proposed algorithms, which can be used to accelerate the computation time.

3.1 Parallel matrix multiplication

One of the most expensive operations in the computation time of the proposed algorithms is the matrix multiplication. Classical matrix multiplication algorithms, for dense matrices and one simple processor, are executed in three nested loops. The use of parallel computation can dramatically improve the time needed to execute this operation. In this work it was used an approach presented in [21]. The general idea is to group the independent operations, run them in parallel and pass the intermediate results sequentially. A schematic of the proposed architecture can be seen in Figure 1. In this example, a solution is presented for two 10 × 10 matrices, which presents a latency of 2 clock pulses (for fixed-point arithmetic). The proposed parallel algorithm presents an excellent complexity property given by \( O(\lceil \log_{10}(n) \rceil + 1) \). For comparison, a multiplication between two 10 × 10 matrices in the parallel algorithm requires 2 pulses of clock, whereas the sequential algorithm requires 1000.

3.2 GPCGPAD implementation architecture

A finite state machine was proposed to implement the GPCGPAD algorithm in a parallel architecture. The solution was centered on the sequential control of the actions in the state machine, so that the intermediate values of computation are coordinated while the operations of addition, subtraction and matrix multiplication execute in parallel. In Figure 2 the implementation of the state machine and the corresponding operations are represented schematically. It can be seen that multiplication operations require 2 pulses of clock, assuming all matrices of dimension 10 × 10, while sums and subtractions are executed in a single pulse. In state 11, the stop criterion is checked and, if it is satisfied, after step 12, the process is then terminated, signalling the control increment signal found. In states 2 and 9, the multiplication is done in only one pulse of clock, since these are multiplications of vectors by scalars. Thus, it is possible to
estimate the execution time of the algorithm as:
\[
t = T_{\text{clock}} \left( 2 + n_{\text{iter}} \left( 8 + 2\left( \left\lceil \log_{10}(N_u) \right\rceil + 1 \right) \right) \right),
\]
where \( T_{\text{clock}} \) is the period of clock used by the hardware, \( n_{\text{iter}} \) is the number of iterations required for the algorithm to reach the stopping criterion, \( N_u \) is the control horizon and \( n_{\text{rin}} \) is the number of inequality constraints.

### 3.3 GPCFAMA implementation architecture

The parallel implementation of GPCFAMA used the same strategy as GPCGPAD and can be seen in Figure 3. In this case, three matrix multiplications depending on \( n_{\text{rin}} \) are required and only one depending on \( N_u \). In state 12, the stopping criterion is verified and, if met, the process is terminated and the control increment signal found is signalled. It is possible to estimate the execution time of the algorithm as:
\[
t = T_{\text{clock}} \left( 2 + n_{\text{iter}} \left( 8 + 2\left( \left\lceil \log_{10}(N_u) \right\rceil + 1 \right) \right) + 3\left( \left\lceil \log_{10}(n_{\text{rin}}) \right\rceil + 1 \right) \right),
\]
with the same definitions of (36).

### 3.4 Preconditioning

A fact that is well known for first-order methods is that preconditioning can offer significant speedup in the convergence rate [28], [10]. In [28], an optimal preconditioning is proposed for FAMA to solve MPC problems with polytopic constraints. The main idea is to enlarge the step size of the algorithm by decreasing \( \lambda_{\text{max}}(R^T R) \). In this paper, a simple preconditioning approach is presented. The first step is to obtain the Cholesky decomposition of the Hessian \( H = KK^T \), which allows (2) to be written as:
\[
\min_{\Delta u} \frac{1}{2} \Delta u^T KK^T \Delta u + b^T \Delta u \\
\text{s.t. } R \Delta u \leq r.
\]

From a variable substitution, \( \Delta u = K^T \Delta u' \), a reshaped problem is obtained:
\[
\min_{\Delta u'} \frac{1}{2} \Delta u'^T K^T (K^T K)^{-1} K^T \Delta u' + \frac{1}{2} b^T (K^T K)^{-1} b \\
\text{s.t. } R (K^T K)^{-1} K^T \Delta u' \leq r.
\]

Although this preconditioning approach is not optimal, it can enlarge the step size in some cases, decreasing the number of iterations. Also, the Hessian becomes an identity matrix, hence the multiplications with \( H \) in the algorithms (step 6 in Figures 2 and 3) do not need to be implemented, which translates into a decrease in the computation time. As an example, a problem with a control horizon \( N_u = 10 \) and with 10 constraints \( (n_{\text{rin}} = 10) \), the computation time can be reduced in approximate 11% per iteration according to equations (36) and (37). It is important to emphasise that the Cholesky decomposition can be computed offline since the matrix \( H \) does not change.

### 4 SIMULATION CASE STUDY

The case study considered in this paper is a three-phase grid-tie LCL-filtered inverter controlling the power injected to or consumed from the grid. These systems have a fundamental role in renewable energy applications and the schematic circuit is shown in Figure 4. PI controllers are widely used in the synchronous rotating \( dq \) frame for regulating the \( d \)- and \( q \)-component currents. However, a PI controller loses its stability when the controller bandwidth is high. Alternatively, proportional-resonant (PR) based control strategy modelled in the stationary \( \alpha\beta \) frame is a solution, but the performance of the PR controller is sensitive to the frequency variations in the grid [29]. Most MPC applications in power converters use the finite control set MPC (FCS-MPC) [30], [31]. In FCS-MPC, the possible control actions are finite and are the switching states. However, the FCS-MPC has a variable switching frequency that brings problems in the quality of the generated signal [18]. Besides, FCS-MPC algorithms use search methods to compute the optimisation and, thus, very small prediction horizons are used (usually 1 or 2). Another method is the continuous control set MPC (CCS-MPC), which has a fixed switching frequency. CCS-MPC has few applications in power...
converter controllers, such as the one presented in [19], [18] and [32], due that the number of calculations needed to solve the constrained optimisation problem on-line, which is incompatible with the small sampling times. Therefore, this case study was chosen for this paper to show how the proposed algorithms can be implemented in real-time and contribute to obtaining satisfactory results with fast computation time.

The dynamical equations of the system (Figure 4) in the synchronously rotating dq-frame are:

\[
\begin{align*}
\frac{di_d}{dt} &= -\frac{r_1}{L_1}i_d - \frac{1}{L_1}v_d + \omega i_q + \frac{1}{L_1}u_d, \\
\frac{di_q}{dt} &= \frac{1}{L_1}i_2 + \frac{1}{L_2}v_d + \omega i_q, \\
\frac{di_2}{dt} &= -\frac{r_2}{L_2}i_d + \frac{1}{L_2}v_d + \omega i_q - \frac{1}{L_2}v_\gamma, \\
\frac{di_\gamma}{dt} &= -\frac{r_1}{L_1}i_q - \frac{1}{L_1}v_d - \omega i_d + \frac{1}{L_1}u_q, \\
\frac{di_\gamma}{dt} &= \frac{1}{C_1}i_q - \frac{1}{C_2}i_2 - \omega i_d, \\
\frac{di_2}{dt} &= -\frac{r_2}{L_2}i_q + \frac{1}{L_2}v_d - \omega i_2 - \frac{1}{L_2}v_\gamma, \\
\end{align*}
\]

where the d- and q-axis components \{i_d, i_q\}, and \{i_2, i_\gamma\} are the inverter currents, and the grid currents, respectively. The capacitor voltages components are \{v_d, v_\gamma\}. The grid voltages are represented by \{v_d, v_\gamma\} and can be assumed as disturbances for the control problem. The components \{u_d, u_q\} denote the voltages before the LCL filter and they are the manipulated variables. The variable \(\omega\) represents the grid frequency in rad/s and the gate signals of the inverter are represented by \{S_v, S_\gamma, S_d, S_\gamma, S_d, S_\gamma\}.

From (40), the discrete-time equivalent transfer matrix used was:

\[
G_{dq}(z) = \begin{bmatrix}
G_d(z) & G_q(z) \\
-G_q(z) & G_d(z)
\end{bmatrix},
\]

where

\[
G_d(z) = \frac{0.0129z^3 + 0.0333z^2 - 0.0148z + 0.0150z^3 - 0.0329z - 0.0127}{z^3 - 1.741z^2 + 2.497z^3 - 3.485z^4 + 2.48z^3 - 1.725z^2 + 0.9802z},
\]

\[
G_q(z) = \frac{0.0004z^3 + 0.0026z^2 - 0.0002z^3 - 0.0026z + 0.0004z^3 - 1.741z^2 + 2.497z^3 - 3.485z^4 + 2.48z^3 - 1.725z^2 + 0.9802z}{z^3 - 1.741z^2 + 2.497z^3 - 3.485z^4 + 2.48z^3 - 1.725z^2 + 0.9802z^2}.
\]

The transfer functions were obtained from the Laplace transform of the dynamic equations. The discretisation was done assuming a zero-order hold model for the inputs and the sample period was chosen as \(T_s = 100\mu s\). The sample period is the same as the switching period used in the PWM. Thus, a time delay of one sample was included in the model to compensate for the computation time delay [31].

A simulation of the current control system of the three-phase inverter was implemented in Simulink with the SimPowerSystems library. The computed control signal, converted in duty-cycle, was used with a carrier-based pulse width modulation (PWM) technique. Also, a two-level bridge based on insulated gate bipolar transistors (IGBTs) was implemented. A phase locked loop (PLL) was used to generate the sine and cosine references from the grid voltages. The simulation circuit parameters are shown in Table 1, where the LCL filter parameters are the same used in [29].

The measured load current was used as a reference for the control system. Thus, the inverter provides the power required by the load. A 5 kW load was used at the beginning of the simulation and, at time \(t = 0.1\) s, another 5 kW load with a reactive portion of 1 kVAR was switched on. The simulation was run for 150 ms. Simulations were performed on a computer with a 2.40 GHz Intel Core i3 processor and 12 GB of RAM. The runtimes were measured using the tic-toc command of MATLAB. The proposed controllers were tuned with control horizons \(N_u = 3\) and \(N_x = 5\), prediction horizons \(N_1 = 3\) and \(N_2 = 5\), weighting in error \(\lambda(1) = \lambda(2) = \lambda = 15\), weighting in error \(\delta(1) = \delta(2) = \delta = 1\), and unknown future references \(u^{(k)}(t+j)\) and \(u^{(k)}(t+j)\). The weights and horizons were defined with equal values because the importance factor and the dynamics of the variables are equivalent.

In abc-frame, the control signal can vary between \([-V_s, +V_s]\), which can easily be represented as a linear inequality constraint in the form \(\mathbf{R}\Delta\mathbf{u} \leq \mathbf{r}\). However, in \(dq\)-coordinates, this constraint becomes a nonlinear inequality: \(\sqrt{u^2_d + u^2_q} \leq \frac{V_s}{\sqrt{3}}\). To use in the algorithms, the following approximation is proposed:

\[
u_d(k+j)u_d(k-1) + u_q(k+j)u_q(k-1) \leq \frac{V_s^2}{3},
\]
with $j = 0, \ldots, \frac{N_u}{2} - 1$. This is a valid approximation if the control signal value does not change too much from one sample to the other. The optimisation problem results in a matrix $H \in \mathbb{R}^{10 \times 10}$ and 10 constraints. The proposed algorithm were compared with an interior point solver from MATLAB (Quadprog), and with the Gurobi solver. The same controller parameters, horizons and cost function were used for all algorithms. The Gurobi solver was tested with dual simplex (GurobiSD) and barrier method (GurobiBar) algorithms [33]. GPCGPAD was tuned for a primal residual tolerance $\epsilon_p = 0.001$ and feasibility tolerance $\epsilon_f = 0.001$. GPCFAMA was tuned with the same primal and dual residual tolerances given by $\epsilon_p = \epsilon_f = \sqrt{\frac{N_u}{2}}0.001$. Gurobi and Quadprog tolerances were also changed to 0.001.

The results can be seen in Figure 5, where the current output of the converter and the control signal applied are presented in the abc reference frame. In Figure 6 the $dq$ components of current output of the converter and the control signal are presented. It can also be seen in this figure that both references are tracked and that the constraints are active and limit the control signal when the setpoint changes. The behavior of the control system was shown only once because it is the same for all the algorithms.

An unconstrained state-space based MPC (uMPC) strategy [18] and a linear quadratic regulator (LQR) [34], which are alternative approaches found in the literature, were implemented. They both use state-space models of the system in the $dq$ reference frame, and the nominal model is augmented with two integrator modes, resulting in a 10-state state-space model, so that the closed-loop system can track step references in each of the $dq$ components. An anti-windup technique was used with both controllers to avoid saturation issues related to the windup of the integrator modes. The control signals profiles are presented in Figure 7 for the uMPC and in Figure 8 for the LQR. The performance index used for comparison was the root mean square error (RMSE). The reference tracking was analysed in $dq$-coordinates and the results can be seen in Table 2. The RMSE obtained for the GPC can be considered equivalent as the one obtained with the uMPC, with the $d$-axis component being 21% smaller for the GPC and the $q$-axis component being 13% smaller for the uMPC. Also, the RMSE obtained from GPC was 2.19 times smaller in the $d$-axis component than the LQR RMSE, and they present approximately the same value for the $q$-axis component. In addition to the good results obtained, the constrained GPC has the advantage of requiring fewer sensors and implicitly avoiding the windup issue.

A comparison between the active and reactive powers at the inverter output and at the load is shown in Figure 9. A load power track can be perceived, which means that the power required in the load is provided by the inverter output. Two Simulink PQ measurement blocks were used to measure the powers at the inverter output and the load.

The performance comparison of the algorithms is illustrated in Figure 10. The execution time of each algorithm is presented in Figure 10(a). It is important to mention that the time is only valid for comparison between the algorithms because the simulation was not done in real time. It is clear that the execution time is shorter for GPCGPAD and GPCFAMA than for Quadprog by more than one order of magnitude and for more than two orders for GurobiSD and GurobiBar. In Figure 10(b) the number of iterations are shown. Even though GPCGPAD and GPCFAMA require a larger number of iterations than the other algorithms during reference changes, they still present shorter computation times, since each iteration is much cheaper from a computational point of view.

Table 3 shows the comparison of the average and maximum execution times, obtained from 10 simulations with each method. From the collected execution time data, the GPCFAMA maximum observed time, with 76 iterations, was 5.90 times faster than Quadprog, 94.69 times faster than GurobiSD, and 84.72 times faster than GurobiBar. When compared with GPCGPAD with 240 iterations, GPCFAMA was 1.43 times faster.
The certification analysis experiment was done to analyse the tightness of the bounds and the increase on the number of iterations with different control and prediction horizons. The certification analysis was applied to the case study problem. The control horizon ranged from $N_u = 5$ to $N_u = 10$ for each control variable and the prediction horizon was kept fixed at $N = 20$ for the output variables.

| TABLE 2 | RMSE comparison for current reference tracking |
| --- | --- |
| | $d$-Axis Component | $q$-Axis Component |
| GPC | 1.64 A | 1.30 A |
| uMPC | 1.99 A | 1.15 A |
| LQR | 3.59 A | 1.29 A |

The theoretical bounds for GPCGPAD were obtained from (18) and the solution of the MILP defined in (21). Experimental results for GPCGPAD can be seen in Figure 11(a). The theoretical bounds for GPCFAMA were obtained from (35) and the sample based estimation approach with 10 000 samples. Experimental results for GPCFAMA can be seen in Figure 11(b). From the experiments it can be seen that the theoretical bounds on the number of iterations increase with the horizons and that they are relatively tight. For GPCGPAD the difference was less than one order of magnitude (4.92 times) and for GPCFAMA was less than 2 orders of magnitude (35.36 times). These theoretical bounds are very important to know, a priori, the WCET. With this information, it is possible to determine if the real-time requirements of the control system can be met.

5 | FPGA IMPLEMENTATION

In order to evaluate the computation time of the algorithms in a high-performance hardware, Algorithms 1 and 2 were implemented in an FPGA. The FPGA used was an Altera, MAX 10 family, part number 10M50DAF484C7G. MAX 10 family presents embedded analog to digital converters and 50 000 programmable logic elements. The on-line optimisation part of the algorithms were implemented directly in a custom logic with a hardware description language called Verilog. To represent the numbers a 32-bit fixed-point arithmetic was used, with 16 bits for the decimals. A state machine was used to synchronise the algorithm and all the matrix operations were implemented for parallel computation.
The worst case observed in the simulation of Section 6 was replicated for implementation in the FPGA. The results obtained for the optimisation computation time can be seen in Table 4. A 50 MHz clock signal was used, and due to the parallelism implemented the execution time is quite fast, with a maximum value for the control signal computation of 76.84 \( \mu s \) for GPAD and 24.36 \( \mu s \) for FAMA. The execution times obtained are consistent with the expected values from (36) and (37). They are compatible with the case study switching frequency, which is 10 kHz even if we add the time used to compute the GPC and the free-response in parallel computation in the same way presented in [21] with 0.14 \( \mu s \) and 0.80 \( \mu s \), respectively. Also, even a switching frequency of 20 kHz can be achieved with the GPCFAMA algorithm. Conversely, the theoretical WCET for both algorithms is not compatible with the sample time. It is important to mention that the theoretical WCET tends to be conservative, and in most cases, the control system can still be used. However, if hard real-time guarantees are required, the FPGA could be changed to another one with faster clock frequency. The Altera Max 10 is a low-cost FPGA, and some advanced solutions, such as Stratix IV, can use a 600 MHz clock frequency, which is 12 times faster.

The results are also compatible with state-space solutions presented in similar hardware, as in [35] where a state-space MPC was used to control the vibration in flexible beam. The plant was modelled with 14 states and a prediction horizon of 14 samples and an interior point method was implemented in an FPGA with 70 MHz of clock frequency, which resulted in an optimisation process executed in 30.0 \( \mu s \). In [36], an MPC with active-set method was applied to control a motor servo system modelled with 2 states. The control horizon used was of 3 samples and the computation time, in an FPGA with 100 MHz, was 20.0 \( \mu s \).

6 | CONCLUSION

This work proposed two fast computation GPC algorithms based on the accelerated dual gradient projection method and the fast alternating minimisation algorithm. Theoretical bounds on the number of iterations for the algorithms were presented.
and are quite important to define the control system components to meet real-time requirements. The proposed algorithms and the commercial solvers considered in this study showed similar results in terms of the resulting control signal, but the proposed methods are at least two orders of magnitude faster when the same hardware is used. The proposed methods allow straightforward implementation in an FPGA because they use only basic operations for on-line computation. The implementation in an FPGA proved to be quite fast for the case study presented and a switching frequency of 10 kHz was considered. Also, even a switching frequency of 20 kHz can be achieved with the GPCFAMA algorithm. These contributions make the constrained GPC suitable for embedded systems and may eventually increase its usage to control processes with fast dynamics.

ACKNOWLEDGEMENTS

This work was supported in part by the National Council for Scientific and Technological Development (CNPq/Brazil), under Grants 304032/2019-0, 305785/2015-0 and 309244/2018-8, in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES/Brazil) under PrInt/UFSC Program, and in part by the Chinese National Natural Science Foundation under Grant 62071124.

ORCID

Vinicius Berndsen Peccin https://orcid.org/0000-0002-1846-4933
Daniel Martins Lima https://orcid.org/0000-0003-3929-8361
Rodolfo César Costa Flesch https://orcid.org/0000-0001-9536-5835
Julio Elias Normey-Rico https://orcid.org/0000-0002-6510-9803

REFERENCES

1. Samad, T.: A survey on industry impact and challenges thereof [technical activities]. IEEE Control Syst. Mag. 37(1), 17–18 (2017)
2. Wang, Y., Boyd, S.: Fast model predictive control using online optimization. IEEE Trans. Control Syst. Technol. 18(2), 267–278 (2010)
3. Bemporad, A., et al.: The explicit linear quadratic regulator for constrained systems. Automatica 38(1), 3–20 (2002)
4. Cai, X., et al.: Fast distributed MPC based on active set method. Comput. Chem. Eng. 71, 158–170 (2014)
5. Ferreau, H.J., et al.: An online active set strategy to overcome the limitations of explicit MPC. Int. J. Robust Nonlinear Control 18(8), 816–830 (2008)
6. Wills, A.G., et al.: Fast linear model predictive control via custom integrated circuit architecture. IEEE Trans. Control Syst. Technol. 20(1), 59–71 (2012)
7. Cairano, S.D., et al.: Projection-free parallel quadratic programming for linear model predictive control. Int. J. Control 86(8), 1367–1385 (2013)
8. Olama, A., et al.: Relaxed hybrid consensus ADMM for distributed convex optimisation with coupling constraints. IET Control Theory Appl. 13(17), 2826–2837 (2019)
9. O’Donoghue, B., et al.: A splitting method for optimal control. IEEE Trans. Control Syst. Technol. 21(6), 2432–2442 (2013)
10. Patrinos, P., Bemporad, A.: An accelerated dual gradient-projection algorithm for embedded linear model predictive control. IEEE Trans. Autom. Control 59(1), 18–33 (2014)
11. Ferreau, H.J., et al.: Embedded optimization methods for industrial automatic control. In: Proceedings of 20th IFAC World Congress, IFAC, New York (2017)
12. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $o(1/k^2)$. Soviet Math. Dokl. 27(2), 372–376 (1983)
13. Patrinos, P., et al.: A dual gradient-projection algorithm for model predictive control in fixed-point arithmetic. Automatica 55, 226–235 (2015)
14. Camacho, E.F., Bordons, C.: Model Predictive Control. In: Advanced Textbooks in Control and Signal Processing. Springer, London (2004)
15. Normey-Rico, J.E., Camacho, E.F.: Control of Dead-time Processes. Springer, London (2007)
16. Judewicz, M.G., et al.: Generalized predictive current control (GPCC) for grid-tie three-phase inverters. IEEE Trans. Ind. Electron. 63(7), 4475–4484 (2016)
17. Judewicz, M.G., et al.: Inverter-side current control of grid-connected voltage source inverters with LCL filter based on generalized predictive control. IEEE J. Emerg. Sel. Topics Power Electron. 6(4), 1752–1743 (2018)
18. Macari, Jr., et al.: Robust model predictive controller applied to three-phase grid-connected LCL filters. J. Control Autom. Electr. Syst. 31(2), 447–460 (2020)
19. Guzman, R., et al.: Receding-horizon model-predictive control for a three-phase VSI with an LCL filter. IEEE Trans. Ind. Electron. 66(9), 6671–6680 (2019)
20. Peccin, V.B., et al.: Fast generalized predictive control based on accelerated dual gradient projection method. In: Proceedings of 12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems (DYCOPS), pp. 474–479. IFAC, New York (2019)
21. Peccin, V.B., et al.: Fast constrained generalized predictive control with ADMM embedded in an FPGA. IEEE Latin America Trans. 18(2), 422–429 (2020)
22. Normey-Rico, J.E., Camacho, E.F.: Multivariable generalised predictive controller based on the Smith predictor. IEEE Proc. Control Theory Appl. 147(5), 538–546 (2000)
23. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
24. Bemporad, A., Patrinos, P.: Simple and certifiable quadratic programming algorithms for embedded linear model predictive control. In: Proceedings of 4th IFAC Nonlinear Model Predictive Control Conference, pp. 14–20. IFAC, New York (2012)
25. Goldstein, T., et al.: Fast alternating direction optimization methods. SIAM J. Imaging Sci. 7(3), 1588–1623 (2014)
26. Stathopoulos, G., et al.: Operator splitting methods in control. Found. Trends Syst. Control 3(3), 249–362 (2016)
27. Combettes, P.L., Pesquet, J.C.: Proximal Splitting Methods in Signal Processing. In: Fixed-Point Algorithms for Inverse Problems in Science and Engineering, pp. 185–212. Springer, New York (2011)
28. Pu, Y., et al.: Complexity certification of the fast alternating minimization algorithm for linear MPC. IEEE Trans. Autom. Control 62(2), 888–893 (2017)
29. Kokot, O., et al.: Model-based current control strategy with virtual time constant for improved dynamic response of three-phase grid-connected VSI. IEEE Trans. Ind. Electron. 66(6), 4156–4165 (2019)
30. Lim, C.S., et al.: Comparison of current control strategies based on FCS-MPC and D-PI-PWM control for actively damped VSCs with LCL-filters. IEEE Access 7, 112410–112423 (2019)
31. Jin, T., et al.: Model predictive voltage control based on finite control set with computation time delay compensation for PV systems. IEEE Trans. Energy Convers. 34(1), 330–338 (2019)
32. Cavanini, L., et al.: Model predictive control for pre-compensated voltage mode controlled DC-DC converters. IET Control Theory Appl. 11(15), 2514–2520 (2017)
33. Gurobi Optimizer Reference Manual. Gurobi Optimization, Inc. (2020)
34. Trajectory-based LQR control of a grid-connected converter with an LCL filter. In: Proceedings of 6th IFAC Conference on Nonlinear Model Predictive Control, pp. 273–278. IFAC, New York (2018)

35. Wills, A., et al. FPGA implementation of an interior-point solution for linear model predictive control. In: Proceedings of 18th IFAC World Congress. IFAC, New York (2011)

36. Yang, N., et al.: Model predictive controller design and implementation on FPGA with application to motor servo system. Control Eng. Practice 20(11), 1229–1235 (2012)

How to cite this article: Peccin VB, Lima DM, Flesch RCC, Normey-Rico JE. Fast algorithms for constrained generalised predictive control with on-line optimisation. IET Control Theory Appl. 2021;15:545–558. https://doi.org/10.1049/cth2.12060