Real-time proximal gradient method for embedded linear MPC[☆]

Ruben Van Parys^{a,*}, Maarten Verbandt^a, Jan Swevers^a, Goele Pipeleers^a

^a*MECO Research Team, Department Mechanical Engineering, KU Leuven,
DMMS corelab, Flanders Make, Leuven, Belgium.*

Abstract

This paper presents and experimentally validates an embedded linear model predictive control (MPC) approach that is particularly useful for implementation on resource-constrained embedded hardware. The MPC scheme is based on a real-time implementation of the proximal gradient method (PGM) and generates input signals with guaranteed constraint satisfaction. Given standard linear MPC assumptions, asymptotic stability of the resulting closed loop is proven. Applied to linear systems with simple input constraints, the real-time PGM results in very simple arithmetics that are rapidly executed on resource-constrained hardware. As a proof on concept, the algorithm is demonstrated on two experimental setups in which it is implemented on a micro-controller. These experimental validations and related simulations demonstrate how the proposed real-time PGM allows fast control rates, especially compared to state-of-the-art linear MPC approaches, and this while preserving good closed-loop performance.

Keywords: linear model predictive control, proximal gradient method, embedded control.

1. Introduction

Model predictive control (MPC) is a popular control approach that relies on the solution of an optimal control problem with a receding horizon [1]. It allows to handle physical system constraints in a systematic manner and often leads to reduced tuning effort and improved control performance. However, evaluating the MPC law is often computationally demanding. It requires solving an optimization problem in real time, which precludes the use of MPC in applications that require fast response times and run on resource-constrained embedded hardware.

For small-scale linearly constrained MPC problems, the computational burden is partially avoided by using an explicit MPC scheme. This is achieved by precomputing the solution map offline using multi-parametric programming, while the online control law consists of region search and table look-up procedures [2, 3]. However, explicit MPC quickly becomes impractical for larger problem sizes, mainly due to substantial memory requirements.

For linear MPC using online optimization, the computational load is minimized by developing efficient strategies to solve the underlying quadratic program (QP). Popular QP solvers exploit the structure of MPC problems and use an active-set method [4], an interior-point method [5, 6] or a dual Newton strategy [7]. Recently, there has been an increased interest in using first-order methods for the online solution of linear MPC problems [8, 9, 10, 11, 12, 13, 14]. In contrast to the aforementioned QP solving strategies, first-order methods do not require the solution of a linear system at every iteration, which is often a limiting factor

[☆]This work benefits from KU Leuven-BOF PFV/10/002 Centre of Excellence: Optimization in Engineering (OPTEC); from the project G0C4515N of the Research Foundation-Flanders (FWO-Flanders); and from the KU Leuven Research project C14/15/067: B-spline based certificates of positivity with applications in engineering. Ruben Van Parys is a PhD fellow of FWO-Flanders. Flanders Make is the Flemish strategic research centre for the manufacturing industry.

*Corresponding author

Email address: rubenvanparys@gmail.com (Ruben Van Parys)

for resource-constrained embedded platforms. Furthermore, first-order methods have certain features that make them amenable to fixed-point implementation and they can be efficiently parallelized. This makes them promising candidates for efficient, low cost MPC.

An intuitive approach to further reduce the computational load of MPC is solving the underlying QP approximately by using a fixed number of iterations. In a next control update, the previous solution is used to warm-start the optimization algorithm to further reduce the suboptimality of the predicted trajectories. This *suboptimal* MPC strategy was already proposed in the 90s [15, 16] and has known many different implementations since then. The use of suboptimal MPC schemes using first-order methods has already been proposed [11, 12, 13, 14]. These approaches mainly focus on finding lower bounds on the number of optimization iterations per control update in order to guarantee a given level of optimality.

This paper proposes a real-time iteration scheme for linear MPC using the proximal gradient method (PGM). The PGM is an extension of the (projected) gradient method and a popular first-order method for linear MPC with simple input constraints [9, 12]. In a real-time iteration scheme, only one PGM iteration is performed per control update, while the next update is warm-started using the solution of the previous one. Applying the real-time PGM scheme to linear systems with simple input constraints, such as box constraints, results in a simple control law with guaranteed constraint satisfaction. This offers possibilities to obtain high control rates on resource-constrained devices such as micro-controllers, field-programmable gate arrays (FPGAs) or programmable logic controllers (PLCs). Under standard assumptions, closed-loop stability is proven for both the system's state and the input trajectory, i.e. over the control updates the system's state is attracted to a stable equilibrium while the suboptimal input trajectory is converging towards its optimal value. In order to retrieve this outcome, current convergence analysis of the PGM is extended with a result relating the monotonic decrease in objective function to the distance to the optimizer. Preliminary results were presented in [17]. The current paper contributes to these results by (i) providing a practical solution for controlling marginally stable and unstable systems, (ii) comparing the presented control approach with state-of-the-art methods for embedded MPC, (iii) adding experimental validations on two different setups in which the real-time PGM is deployed on a resource-constrained micro-controller and (iv) providing a software package that facilitates the design and code generation of the proposed control strategy.

Section 2 describes the linear MPC problem addressed in this work, while Section 3 introduces the proximal gradient method. Section 4 presents the real-time PGM scheme and Section 5 proves asymptotic stability of the resulting closed-loop system. Thereafter, the closed-loop performance is validated in simulation and via experiments in Section 6. Finally, Section 7 draws concluding remarks.

2. Problem description

This work focuses on the control of a discrete-time linear time-invariant system of the form

$$x_{k+1} = Ax_k + Bu_k, \quad (1)$$

with current and next state $x_k, x_{k+1} \in \mathbb{R}^n$ and input $u_k \in \mathbb{R}^m$. The input is confined to a compact convex set $\mathbb{U} \subset \mathbb{R}^m$, which typically represents actuator limitations. Given a measurement or estimate of the current state x_k , an MPC controller will compute an input trajectory $q = \{q(0), \dots, q(N-1)\}$ and state trajectory $s = \{s(0), \dots, s(N)\}$ over a control horizon of N samples. Optimal trajectories are retrieved by solving the following optimal control problem:

$$\begin{aligned} \min_{q, s} \quad & \sum_{i=0}^{N-1} l(s(i), q(i)) + V_f(s(N)) \\ \text{s.t.} \quad & s(0) = x_k, \\ & s(i+1) = As(i) + Bq(i), \forall i \in \{0, \dots, N-1\}, \\ & q(i) \in \mathbb{U}, \forall i \in \{0, \dots, N-1\}, \\ & s(N) \in \mathbb{X}_f. \end{aligned} \quad (2)$$

The set $\mathbb{X}_f$ is a convex compact set representing a possible terminal state constraint to guarantee closed-loop stability. With $q^* = \{q^*(x_k, 0), \dots, q^*(x_k, N-1)\}$ the optimal solution of (2), an optimal MPC approach will apply the control input $u_k = q^*(x_k, 0)$.

Similar as in a classical linear MPC formulation, the following assumption is made:

Assumption 1. *The functions l and V_f are quadratic and of the following form:*

$$\begin{aligned} l(x, u) &= \frac{1}{2}x^T Qx + \frac{1}{2}u^T Ru, \\ V_f(x) &= \frac{1}{2}x^T Px, \end{aligned}$$

where $Q \in \mathbb{R}^{n \times n}$, $R \in \mathbb{R}^{m \times m}$ and $P \in \mathbb{R}^{n \times n}$ are positive definite, i.e. $Q \succ 0$, $R \succ 0$, $P \succ 0$. Furthermore, P and $\mathbb{X}_f$ are chosen such that $\forall x \in \mathbb{X}_f, \exists \kappa_f(x) \in \mathbb{U}$:

$$(i) \quad Ax + B\kappa_f(x) \in \mathbb{X}_f, \quad (3a)$$

$$(ii) \quad V_f(Ax + B\kappa_f(x)) - V_f(x) \leq -l(x, \kappa_f(x)). \quad (3b)$$

Assumption 1 is a necessary condition to prove exponential stability of the closed-loop system when using an optimal MPC law [1]. The goal of this paper is to prove closed-loop stability when using a suboptimal solution of (2) obtained by only one iteration of the proximal gradient method.

3. Proximal gradient method

This section describes the proximal gradient method (PGM), which composes the underlying first-order optimization strategy of the presented MPC approach. First, the problem family suited for the PGM is presented. Then, the PGM is described and a relation is derived between the monotonic decrease in objective value during each PGM step and the distance to the optimizer. This relation is a key element in the overall closed-loop stability proof of Section 5.

3.1. Problem description

The proximal gradient method is used to solve problems of the following form

$$\min_q h(q) = f(q) + g(q), \quad (4)$$

where the objective function h is split in two parts, f and g , that satisfy the following assumption:

Assumption 2. *The function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is closed and strongly convex with parameter $m > 0$ and ∇f is Lipschitz continuous with parameter L . The function $g : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ is proper, closed, convex and extended real-valued, i.e. $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$.*

Strong convexity of f implies a quadratic lower bound on f [18]:

$$f(p) \geq f(q) + \nabla f(q)^T(p - q) + \frac{m}{2}\|p - q\|_2^2 \quad \forall q, p \in \mathbb{R}^d,$$

and strong monotonicity of ∇f :

$$(\nabla f(q) - \nabla f(p))^T(q - p) \geq m\|q - p\|_2^2 \quad \forall q, p \in \mathbb{R}^d. \quad (5)$$

Using the Cauchy-Schwarz inequality, (5) implies

$$\|\nabla f(q) - \nabla f(p)\|_2 \geq m\|q - p\|_2 \quad \forall q, p \in \mathbb{R}^d. \quad (6)$$

Lipschitz continuity of ∇f implies a *quadratic upper bound* on f [18]:

$$f(p) \leq f(q) + \nabla f(q)^T(p - q) + \frac{L}{2}\|p - q\|_2^2 \quad \forall q, p \in \mathbb{R}^d, \quad (7)$$

and *co-coercivity* of ∇f :

$$(\nabla f(q) - \nabla f(p))^T(q - p) \geq \frac{1}{L}\|\nabla f(q) - \nabla f(p)\|_2^2 \quad \forall q, p \in \mathbb{R}^d. \quad (8)$$

As g is possibly non-smooth, the analysis in Section 3.3 can only rely on the definition of a subgradient [18]: $w \in \mathbb{R}^d$ is a subgradient of g at $q \in \mathbb{R}^d$ if

$$g(p) \geq g(q) + w^T(p - q) \quad \forall p \in \mathbb{R}^d. \quad (9)$$

The set of all subgradients of g at q is the subdifferential $\partial g(q)$.

3.2. The proximal gradient method

The proximal gradient method belongs to the set of proximal algorithms [19], a group of methods that use the *proximal mapping* as base operation. The proximal mapping prox_g of a function g is defined as

$$\tilde{p} = \text{prox}_g(q) = \underset{p}{\operatorname{argmin}} g(p) + \frac{1}{2}\|p - q\|_2^2, \quad (10)$$

i.e. $\text{prox}_g(q)$ returns a point $\tilde{p}$ that makes a trade-off between minimizing g and staying close to q . Following from the first-order optimality conditions, (10) is equivalent to

$$q \in \tilde{p} + \partial g(\tilde{p}). \quad (11)$$

When g is proper, closed and convex, the point $\tilde{p}$ exists and is unique, i.e. prox_g is single-valued. Furthermore, proximal mappings are *nonexpansive*:

$$\|\text{prox}_g(q) - \text{prox}_g(p)\|_2 \leq \|q - p\|_2. \quad (12)$$

Suppose g is the indicator of a closed convex set $\mathcal{Q}$, i.e.

$$g(q) = \mathcal{I}_{\mathcal{Q}}(q) = \begin{cases} 0, & q \in \mathcal{Q}, \\ +\infty, & q \notin \mathcal{Q}, \end{cases}$$

then the proximal mapping of g represents the projection on $\mathcal{Q}$, i.e. $\text{prox}_{\mathcal{I}_{\mathcal{Q}}}(q) = \text{proj}_{\mathcal{Q}}(q)$.

The proximal gradient method solves problem (4) iteratively, by performing sequentially a forward gradient step on f and a backward subgradient step on g :

$$q^+ \in (q - \gamma \nabla f(q)) - \gamma \partial g(q^+). \quad (13)$$

Using (11), this is equivalently written using the prox_g operator:

$$\begin{aligned} q^+ &= \text{prox}_{\gamma g}(q - \gamma \nabla f(q)), \\ &= q - \gamma G_{\gamma}(q), \end{aligned} \quad (14)$$

where $G_{\gamma}(q) = \frac{q - q^+}{\gamma}$ is an analogue of the usual gradient to nonsmooth functions of the form (4) [20]. When $\gamma \in (0, \frac{2}{L})$ and when Assumption 2 holds, the method converges to the optimizer q^* of (4) [20].

3.3. Monotonic decrease in objective value

This subsection derives a relation between the monotonic decrease in the objective function of (4) due to one PGM step (14) and the distance to the optimizer. This is summarized in the following theorem.

Theorem 1. *Consider a function $h = f + g$, where f and g satisfy Assumption 2. Suppose $q \in \mathbb{R}^d$ is mapped on $q^+ \in \mathbb{R}^d$ using the PGM step (14) with $\gamma \in (0, \frac{2}{L})$ and q^* is the minimizer of h . Then, there exists a $\beta > 0$ such that*

$$h(q^+) - h(q) \leq -\beta \|q - q^*\|_2^2.$$

Proof. First the PGM step (14) is shown to reduce the distance to the optimizer:

$$\begin{aligned} \|q^+ - q^*\|_2^2 &= \|\text{prox}_{\gamma g}(q - \gamma \nabla f(q)) - \text{prox}_{\gamma g}(q^* - \gamma \nabla f(q^*))\|_2^2, \\ &\leq \|q - \gamma \nabla f(q) - q^* + \gamma \nabla f(q^*)\|_2^2, \\ &\leq \|q - q^*\|_2^2 - \gamma \left(\frac{2}{L} - \gamma\right) \|\nabla f(q) - \nabla f(q^*)\|_2^2, \\ &\leq \underbrace{\left(1 - \frac{2m^2}{L}\gamma + m^2\gamma^2\right)}_{\varepsilon^2} \|q - q^*\|_2^2. \end{aligned} \tag{15}$$

The first inequality follows from the nonexpansiveness of the proximal mapping (12). The second inequality follows from co-coercivity of ∇f (8) and the third from strong monotonicity of ∇f (6). Since $\gamma \in (0, \frac{2}{L})$, it follows that $(1 - \frac{2m^2}{L}\gamma + \gamma^2 m^2) \in (0, 1)$.

Inequality (15) allows us to find a relation between $\|G_\gamma(q)\|_2^2$ and the distance to the optimum:

$$\begin{aligned} \gamma^2 \|G_\gamma(q)\|_2^2 &= \|(q - q^*) - (q^+ - q^*)\|_2^2, \\ &\geq (\|q - q^*\|_2 - \|q^+ - q^*\|_2)^2, \\ &\geq (1 - \varepsilon)^2 \|q - q^*\|_2^2. \end{aligned} \tag{16}$$

The first inequality is the reverse triangle inequality and the second inequality follows from (15).

Finally a relation between the decrease in objective value and the distance to the optimizer is derived:

$$\begin{aligned} h(q^+) - h(q) &= f(q^+) - f(q) + g(q^+) - g(q), \\ &\leq \nabla f(q)^T (q^+ - q) + \frac{L}{2} \|q^+ - q\|_2^2 + g(q^+) - g(q), \\ &\leq \nabla f(q)^T (q^+ - q) + \frac{L}{2} \|q^+ - q\|_2^2 + (G_\gamma(q) - \nabla f(q))^T (q^+ - q), \\ &= -\gamma \left(1 - \frac{L}{2}\gamma\right) \|G_\gamma(q)\|_2^2, \\ &\leq \underbrace{-\frac{1}{\gamma} \left(1 - \frac{L}{2}\gamma\right) (1 - \varepsilon)^2}_{\beta} \|q - q^*\|_2^2. \end{aligned}$$

The first inequality expresses the quadratic upper bound (7) for f due to Lipschitz continuity of ∇f . The second inequality follows from (13) which implies that $G_\gamma(q) - \nabla f(q) \in \partial g(q^+)$ and from the definition of a subgradient (9). The last inequality follows from (16). Since $\gamma \in (0, \frac{2}{L})$, it follows that $\beta > 0$. $\square$

Algorithm 1 Real-time proximal gradient method

Require: $\gamma \in (0, \frac{2}{L})$

- 1: **Repeat** every sample k
 - 2: Retrieve state estimate x_k
 - 3: $q_k^+ = \text{proj}_{\mathcal{Q}(x_k)}(q_k - \gamma \nabla_q V_N(x_k, q_k))$
 - 4: Apply $q_k^+(0)$ to the system
 - 5: $q_{k+1} = \{q_k^+(1), \dots, q_k^+(N-1), \kappa_f(\phi(N, x_k, q_k^+))\}$
 - 6: **End**
-

4. Real-time proximal gradient method MPC

4.1. Real-time PGM algorithm

After eliminating the state trajectory s and using Assumption 1, optimal control problem (2) is rewritten as the following problem:

$$\min_{q \in \mathcal{Q}(x_k)} V_N(x_k, q) = \frac{1}{2} q^T F q + \frac{1}{2} x_k^T G x_k + q^T H x_k, \quad (18)$$

where the input trajectory q is the decision variable and the current state x_k is a parameter for the problem. The feasible set $\mathcal{Q}(x_k)$ is determined from the input bounds and terminal state constraint:

$$\mathcal{Q}(x_k) = \{q \mid q(i) \in \mathbb{U}, i = 0, \dots, N-1, \phi(N, x_k, q) \in \mathbb{X}_f\}.$$

Here, $\phi(N, x_k, q)$ represents the terminal state following from x_k, q and the integration of (1):

$$\phi(N, x_k, q) = A^N x_k + \sum_{i=0}^{N-1} A^{N-1-i} B q(i). \quad (19)$$

Using an indicator function, problem (18) is equivalently rewritten:

$$\min_q h(x_k, q) = V_N(x_k, q) + \mathcal{I}_{\mathcal{Q}(x_k)}(q). \quad (20)$$

By Assumption 1, F is positive definite and therefore $V_N(x_k, \cdot)$ is strongly convex with parameter the minimum eigenvalue of F , i.e. $m = \text{eig}_{\min}(F)$. Furthermore, $\nabla_q V_N(x_k, \cdot)$ is Lipschitz continuous with Lipschitz constant the maximum eigenvalue of F , i.e. $L = \text{eig}_{\max}(F)$. The function $\mathcal{I}_{\mathcal{Q}(x_k)}(\cdot)$ is closed and convex as $\mathcal{Q}(x_k)$ is compact and convex. Problem (20) follows the structure of (4) where $V_N(x_k, \cdot)$ represents the smooth part and $\mathcal{I}_{\mathcal{Q}(x_k)}(\cdot)$ the non-smooth part of the objective function.

This work proposes an MPC scheme that implements a real-time version of the PGM to solve optimal control problem (20) in receding horizon. This is demonstrated in Algorithm 1. Only one PGM step (14) is performed per control update (line 3). The first sample of the resulting input trajectory is applied to the system (1) (line 4) and a warm-start for the next update is computed (line 5). This is achieved by shifting the input trajectory over one sample and expanding its last sample using the control law $\kappa_f(\cdot)$, introduced in Assumption 1, applied to the final state. Note that, although Algorithm 1 presents a suboptimal MPC approach, the applied input $q_k^+(0)$ is guaranteed to satisfy the imposed constraints, as q_k^+ is the result of a projection on the feasible set $\mathcal{Q}(x_k)$.

4.2. Practical implementation

The real-time PGM algorithm is only useful in practice when the projection $\text{proj}_{\mathcal{Q}(x_k)}$ is easy to compute. The inclusion of state constraints, such as the terminal set $\mathbb{X}_f$, in $\mathcal{Q}(x_k)$ generally complicates this requirement. Existing approaches to omit this terminal set use soft constraining [21] or require the horizon length N to be sufficiently large [22]. Other approaches dualize the terminal state constraint and use a dual optimization strategy [11, 9, 13]. All these results however rely on solving the OCPs until optimality and are therefore unusable in a suboptimal setting where only one PGM step is executed. In order to guarantee closed-loop stability for suboptimal MPC, the terminal set $\mathbb{X}_f$ is typically included explicitly. In order to show how this is achieved in practice, a distinction is made between stable and unstable systems.

4.2.1. Stable systems

For stable linear systems, that were considered in [17], assumptions (3a) and (3b) hold for the following choice of terminal set, terminal control law and terminal cost:

- (i) $\mathbb{X}_f = \mathbb{R}^n$,
- (ii) $\kappa_f(\cdot) = 0$,
- (iii) $V_f(x) = \frac{1}{2}x^T P x$, where P is the solution to the Lyapunov equation $A^T P A + Q = P$.

As the terminal state constraint can be omitted, $\mathcal{Q}(x_k) = \mathbb{U} \times \dots \times \mathbb{U}$ and it remains to restrict $\mathbb{U}$ to sets that allow a simple projection. Examples are the m -dimensional Euclidean ball, the positive orthant, the simplex, box, hyperplane, halfspace, the second order cone and the positive definite cone [19]. For example, when $\mathbb{U} = [u_{\min}, u_{\max}]$, the projection on $\mathcal{Q}(x_k)$ represents the saturation function, i.e. $\text{proj}_{\mathcal{Q}(x_k)} = \text{proj}_{\mathbb{U} \times \dots \times \mathbb{U}} = \text{sat}_{u_{\min}}^{u_{\max}}$, where the latter is defined as:

$$\text{sat}_{u_{\min}}^{u_{\max}}\{q\}(k) = \begin{cases} u_{\min}, & q(k) < u_{\min}, \\ q(k), & u_{\min} \leq q(k) \leq u_{\max}, \\ u_{\max}, & q(k) > u_{\max}, \end{cases}$$

i.e., each element of q is thresholded at the boundary of $[u_{\min}, u_{\max}]$.

4.2.2. Unstable and marginally stable systems

Unstable and marginally stable linear discrete-time systems have an A matrix with at least one eigenvalue with norm greater than or equal to 1. For those systems, a restrictive terminal state constraint is necessary to guarantee closed-loop stability. To set up this constraint, a real Schur decomposition of A is employed:

$$A = \begin{bmatrix} S_s & S_u \end{bmatrix} \begin{bmatrix} A_s & * \\ 0 & A_u \end{bmatrix} \begin{bmatrix} S_s^T \\ S_u^T \end{bmatrix},$$

in which $A_s \in \mathbb{R}^{n_s \times n_s}$ and $A_u \in \mathbb{R}^{n_u \times n_u}$ are quasi-upper triangular matrices containing the stable and unstable eigenvalues of A respectively, and S_s and S_u are orthogonal matrices. In order to obtain a projection $\text{proj}_{\mathcal{Q}(x_k)}$ which is relatively easy to compute and in order to meet assumptions (3a) and (3b), the terminal set, terminal control law and terminal cost are chosen as:

- (i) $\mathbb{X}_f = \{x \in \mathbb{R}^n \mid S_u^T x = 0\}$,
- (ii) $\kappa_f(\cdot) = 0$,
- (iii) $V_f(x) = \frac{1}{2}x^T S_s P_s S_s^T x$, where P_s is the solution to the Lyapunov equation $A_s^T P_s A_s + S_s^T Q S_s = P_s$.

The terminal constraint set $\mathbb{X}_f$ zeroes the unstable modes at the end of the control horizon. As the terminal state is expressed as an affine function in the input trajectory q (19), the terminal state constraint represents an affine constraint in q :

$$\phi(N, x_k, q) \in \mathbb{X}_f \iff Jq + Kx_k = 0,$$

where $J \in \mathbb{R}^{n_u \times Nm}$ and $K \in \mathbb{R}^{n_u \times n}$ follow from multiplying S_u^T by the matrices of expression (19).

The feasible set $\mathcal{Q}(x_k)$ thus represents the intersection of an affine set and the input constraint set $\mathbb{U} \times \dots \times \mathbb{U}$. As $\text{proj}_{\mathcal{Q}(x_k)}(q)$ is the optimizer of the following problem

$$\begin{aligned} & \underset{y \in \mathbb{U} \times \dots \times \mathbb{U}}{\text{minimize}} && \frac{1}{2} \|y - q\|_2^2 \\ & \text{subject to} && Jy + Kx_k = 0, \end{aligned} \tag{21}$$

it should minimize the corresponding Lagrangian:

$$\begin{aligned}\text{proj}_{\mathcal{Q}(x_k)}(q) &= \underset{y \in \mathbb{U} \times \dots \times \mathbb{U}}{\text{argmin}} \frac{1}{2} \|y - q\|_2^2 + \mu^T (Jy + Kx_k), \\ &= \underset{y \in \mathbb{U} \times \dots \times \mathbb{U}}{\text{argmin}} \frac{1}{2} \|y - q + J^T \mu\|_2^2, \\ &= \text{proj}_{\mathbb{U} \times \dots \times \mathbb{U}}(q - J^T \mu).\end{aligned}$$

In case $\mathbb{U}$ allows an easy projection, it remains to find the optimal Lagrange multiplier $\mu \in \mathbb{R}^{n_u}$ in order to compute $\text{proj}_{\mathcal{Q}(x_k)}(q)$. The multiplier is the result of the following nonlinear root-finding problem, which follows from primal feasibility of (21):

$$w(\mu) = J \text{proj}_{\mathbb{U} \times \dots \times \mathbb{U}}(q - J^T \mu) + Kx_k = 0. \quad (22)$$

In practice it is verified that this problem is easily solved using a quasi-Newton method such as the Broyden's method [23].

Algorithm 2 presents the real-time PGM scheme for possible unstable linear discrete-time systems with box input constraints, i.e. $\mathbb{U} = [u_{\min}, u_{\max}]$. Line 3 represents the gradient step on V_N while lines 4 - 20 compute $\text{proj}_{\mathcal{Q}(x_k)}$. This is achieved by employing the Broyden's method to find the multiplier μ as the root of w (22). The Broyden's method iteratively updates the matrix $Z \in \mathbb{R}^{n_u \times n_u}$ (line 16), which represents an approximation of the inverse of the Jacobian of w . On line 17, the current update Z_j is used to perform a Newton step on the current multiplier μ_j . The Broyden's method starts from an initial value for Z (line 12) and two initial values for μ (line 4 and 8). The latter are chosen as possible good guesses that allow to skip further Broyden iterations: μ_0 and μ_1 are determined such that $\tilde{q} - J^T \mu$ corresponds to respectively $\tilde{q}$ itself and the projection of $\tilde{q}$ on the affine set. The multiplier which zeroes (22) is used to compute q_k^+ on line 20.

Algorithm 2 consists of simple arithmetics which are implementation-friendly for resource-constrained embedded hardware. It involves a matrix-vector multiplication on line 3 with a complexity which mainly depends on the horizon length N . The Broyden iterations involve saturation, scalar division and matrix-vector multiplications with a complexity determined by n_u , the number of unstable modes. In practice, and as illustrated in Section 6, the number of required Broyden's iterations is rather limited and reduces over the control updates due to the warm-starting of q_k . Especially when $n_u \ll N$, the projection step typically requires much less time than the gradient step. Recall that for stable systems, the projection reduces to a single evaluation of the saturation function.

5. Closed-loop stability of the Real-time PGM

When applying the real-time PGM to the system (1), the closed-loop dynamics are written as:

$$\begin{aligned}q_k^+ &= \text{proj}_{\mathcal{Q}(x_k)}(q_k - \gamma \nabla_q V_N(x_k, q_k)), \\ x_{k+1} &= Ax_k + Bq_k^+(0), \\ q_{k+1} &= \{q_k^+(1), \dots, q_k^+(N-1), \kappa_f(\phi(N, x_k, q_k^+))\}.\end{aligned} \quad (23)$$

This closed-loop system has as states the system state x_k and the input trajectory q_k . This section proves asymptotic stability of (23), based on standard MPC assumptions and standard convergence assumptions for the PGM.

Theorem 2. *Under Assumption 1, when $\gamma \in (0, \frac{2}{L})$ and with $\mathcal{X} = \{x \mid \mathcal{Q}(x) \neq \emptyset\}$, the set $\mathcal{X} \times \mathcal{Q}(\mathcal{X})$ is positive invariant for the closed-loop system (23) and the function $h(x, q)$ is a Lyapunov function for (23), satisfying for all $(x, q) \in \mathcal{X} \times \mathcal{Q}(\mathcal{X})$*

$$\begin{aligned}h(x_{k+1}, q_{k+1}) - h(x_k, q_k) &\leq -\alpha_1(\|(x_k, q_k - q^*(x_k))\|), \\ h(x, q) &\geq \alpha_2(\|(x, q - q^*(x))\|), \\ h(x, q) &\leq \alpha_3(\|(x, q - q^*(x))\|),\end{aligned}$$

Algorithm 2 Real-time PGM for linear systems with box input constraints

Require: $\gamma \in (0, \frac{2}{\text{eig}_{\max}(F)})$, $\epsilon > 0$

```

1: Repeat every sample  $k$ 
2:   Retrieve state estimate  $x_k$ 
3:    $\tilde{q}_k = (I - \gamma F)q_k - \gamma Hx_k$ 
4:    $\mu_0 = 0$ 
5:   if  $\|w(\mu_0)\|_2^2 \leq \epsilon$  then
6:     go to 20
7:   end if
8:    $\mu_1 = (JJ^T)^{-1}(J\tilde{q}_k + Kx_k)$ 
9:   if  $\|w(\mu_1)\|_2^2 \leq \epsilon$  then
10:    go to 20
11:  end if
12:   $Z_0 = I_{n_u}$ 
13:   $j = 1$ 
14:  while  $\|w(\mu_j)\|_2^2 > \epsilon$  do
15:     $\Delta\mu_j = \mu_j - \mu_{j-1}$ ,  $\Delta w_j = w(\mu_j) - w(\mu_{j-1})$ 
16:     $Z_j = Z_{j-1} + \frac{\Delta\mu_j - Z_{j-1}\Delta w_j}{\|\Delta w_j\|_2^2} \Delta w_j^T$ 
17:     $\mu_{j+1} = \mu_j - Z_j w(\mu_j)$ 
18:     $j = j + 1$ 
19:  end while
20:   $q_k^+ = \text{sat}_{u_{\min}}^{u_{\max}}\{\tilde{q}_k - J^T \mu_j\}$ 
21:  Apply  $q_k^+(0)$  to the system
22:   $q_{k+1} = \{q_k^+(1), \dots, q_k^+(N-1), 0\}$ 
23: End

```

in which $\alpha_1, \alpha_2, \alpha_3$ are $\mathcal{K}_\infty$ functions (i.e. $\alpha_i : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ and is continuous, strictly increasing, zero at zero and unbounded). Therefore it follows that $(0, q^*(0))$ is asymptotically stable for states (x, q) of system (23) with a region of attraction $\mathcal{X} \times \mathcal{Q}(\mathcal{X})$.

Proof. Because $\mathcal{Q}(x)$ is invariant under the projection $\text{proj}_{\mathcal{Q}(x)}$ and because $\forall x \in \mathbb{X}_f, \exists \kappa_f(x) \in \mathbb{U}$ such that (3a) holds, the set $\mathcal{X} \times \mathcal{Q}(x)$ is positive invariant for (23).

A decrease in h is shown as follows:

$$\begin{aligned}
 h(x_{k+1}, q_{k+1}) &\leq h(x_k, q_k^+) - l(x_k, q_k^+(0)), \\
 &\leq h(x_k, q_k) - l(x_k, q_k^+(0)) - \beta \|q_k - q^*(x_k)\|_2^2, \\
 &\leq h(x_k, q_k) - \alpha_1(\|(x_k, q_k - q^*(x_k))\|),
 \end{aligned}$$

where the first inequality follows from assumption (3b). The second inequality follows from Theorem 1. The third inequality is true as l is positive definite and $\beta > 0$.

A lower bound on h is found for all (x, q) :

$$\begin{aligned}
 h(x, q) &\geq V_N(x, q^*(x)) + \frac{m}{2} \|q - q^*(x)\|_2^2, \\
 &\geq l(x, q^*(x)) + \frac{m}{2} \|q - q^*(x)\|_2^2, \\
 &\geq \alpha_2(\|(x, q - q^*(x))\|),
 \end{aligned}$$

where the first inequality follows from strong convexity of $V_N(x, \cdot)$, the subgradient inequality (9) and the optimality condition $-\nabla_q V_N(x, q^*(x)) \in \partial \mathcal{I}_{\mathcal{Q}(x)}(q^*(x))$. The second inequality follows from the fact that $V_N(x, q^*(x)) \geq l(x, q^*(x))$ [1]. The third inequality follows from positive definiteness of l and from $m > 0$.

An upper bound on h is found for all $(x, q) \in \mathcal{X} \times \mathcal{Q}(\mathcal{X})$. In this set, the indicator $\mathcal{I}_{\mathcal{Q}(x)}$ evaluates to 0 and the upper bound is established as follows:

$$\begin{aligned} h(x, q) &= V_N(x, q) \\ &\leq V_N(x, q^*(x)) + \nabla_q V_N(x, q^*(x))^T (q - q^*) + \frac{L}{2} \|q - q^*\|_2^2, \\ &\leq V_N(x, q^*(x)) + \|\nabla_q V_N(x, q^*(x))\|_2 \|q - q^*\|_2 + \frac{L}{2} \|q - q^*\|_2^2, \\ &\leq \alpha_3(\|(x, q - q^*(x))\|), \end{aligned}$$

where the first inequality follows from Lipschitz continuity of $\nabla_q V_N$. The second inequality follows from the Cauchy-Schwarz inequality. The third inequality is true as $L > 0$ and because V_f allows a $\mathcal{K}_\infty$ upper bound [1].

The aforementioned derivations prove that $h(x, q)$ is a Lyapunov function for the system (23) and therefore it follows that $(0, q^*(0))$ is asymptotically stable with region of attraction $\mathcal{X} \times \mathcal{Q}(\mathcal{X})$ [1]. $\square$

Remark 1. Theorem 2 can be extended for Q positive semi-definite, i.e. $Q \succeq 0$, which occurs when the stage cost l is applied to an output $z = Cx$ and when C is not invertible. Then, an assumption on detectability or input/output-to-state stability should hold which ensures that the state goes to zero as the output and input go to zero [1]. Positive semi-definiteness of Q does however not harm the positive definiteness of F and the strong convexity of $V_N(x, \cdot)$.

Remark 2. Theorem 2 not only shows that the system's state x is attracted to its stable equilibrium but also guarantees that the suboptimal trajectory q converges towards the optimal trajectory $q^*(x)$.

6. Numerical and experimental results

The performance of the presented real-time PGM is validated on two experimental example cases. The first one considers the control of a cart to suppress swinging of a pendulum attached to it. The second case considers the position control of a ball rolling over a tilting table. The following subsections elaborate the cases in detail.

6.1. Pendulum control

6.1.1. Setup overview

The performance of the presented real-time PGM is validated on the experimental setup presented in Figure 1. The setup, which was developed for educational purposes [24], is equipped with two DC motors with encoders, that drive the front wheels. The rear wheels are mounted on an axle held by the rear inserts to constrain the platform to linear motions. A pendulum is attached to the setup with a potentiometer that measures the pendulum angle. The setup is equipped with an Arduino MEGA 2560 micro-controller which reads out the sensors and controls the actuators.

On the Arduino, a velocity controller for the motors is implemented such that the system input u becomes the reference velocity for the cart. The following model relates the input u to the cart position z and pendulum angle θ :

$$l\ddot{\theta} + c\dot{\theta} + g\sin(\theta) + \frac{1}{\tau}\dot{z}\cos(\theta) = \frac{1}{\tau}\cos(\theta)u. \quad (24)$$

Here, l and c represent the length and damping coefficient of the pendulum respectively, while g is the gravitational acceleration and τ denotes the time constant of the internal velocity control loop. The pendulum mass position z_p is computed as $z_p = z - l\sin(\theta)$. After linearizing the system dynamics around the stable

Figure 1: Experimental setup used to validate the real-time PGM.

g	c	l	τ
9.81 m/s ²	0.20 m/s	0.13 m	0.10 s

Table 1: Model parameters of the experimental setup.

pendulum orientation $\theta = 0$ rad, and when choosing the states as $x = [z, \dot{z}, \theta, \dot{\theta}]^T$, the continuous-time dynamics are described by the following state-space equations:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{\tau} & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{1}{l\tau} & -\frac{g}{l} & -\frac{c}{l} \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{1}{\tau} \\ 0 \\ \frac{1}{l\tau} \end{bmatrix} u, \quad (25)$$

$$z_p = [1 \quad 0 \quad -l \quad 0] x,$$

The identified model parameters are summarized in Table 1. The model (25) is discretized using zero-order hold with sample frequency 50 Hz. A linear quadratic estimator is implemented on the Arduino to estimate the system's state.

6.1.2. Simulation results

In a first set of experiments, an MPC controller is developed to track a constant reference r by the pendulum mass position z_p . This is achieved by using the following stage cost:

$$l(x, u) = \rho(r - z_p)^2 + u^2.$$

In this way, pendulum swinging is suppressed when repositioning the cart. The term u^2 is included such that Assumption 1 holds with $R \succ 0$. By giving a large weight to the first term, i.e. $\rho = 1e5 \text{ 1/s}^2$, reference tracking is favored over input limitation. This is acceptable as hard actuator bounds are included: $u \in [-0.3 \text{ m/s}, 0.3 \text{ m/s}]$. Because the dynamics (25) contain an integrator, a terminal state constraint should be included and the Broyden's method from Algorithm 2 is used to compute the projection on the feasible set.

First the response of the real-time PGM is evaluated in simulation using (24) as simulation model. A notebook is utilized with Intel Core i5-4300M CPU with four cores at 2.60 GHz and 8 GB of RAM. For a reference $r = 0.12 \text{ m}$ and a horizon length $N = 20$, the generated input sequence u and the resulting trajectories for z_p , z and θ are depicted in Figure 2. It is clear how the real-time PGM method generates bounded inputs u to steer the system to its equilibrium state. Figure 2 also shows the number of required Broyden iterations n_B to solve (22) with a tolerance $\epsilon = 1e-8$. The number of iterations is limited and decreases over the control updates due to warm-starting the updates.

The response of the real-time PGM is compared to the one obtained by an optimal MPC approach, i.e. one in which the optimal control problem (2) is solved until optimality during each control update. Therefore

Figure 2: Simulated step response for the pendulum mass ($N = 20$). The black and gray lines indicate the response using the real-time PGM and optimal MPC respectively.

two state-of-the-art QP solvers for embedded MPC are used: qpOASES [4], which uses an active-set strategy, and HPMPC [6], which uses an interior-point method. Although the optimal MPC approach generates a more optimal looking bang-bang input sequence (Figure 2), the resulting trajectory for z_p is similar to the one generated by the real-time PGM. The real-time PGM’s relative increase in infinite-horizon objective value with respect to optimal MPC is only 0.80 %. When comparing the solving time t_s for each control update, the optimal approaches require on average 200 to 400 times more time to evaluate the control law. This demonstrates that the real-time PGM offers a computationally cheap alternative for optimal MPC with little loss in performance.

Figure 3 investigates the influence of the horizon length N on the performance of the real-time PGM. As the complexity of the gradient step scales quadratically with N , one could expect an increase in solving time t_s and a decrease in the portion of the time t_{proj} to compute $\text{proj}_{\mathcal{Q}(x_k)}$. The overall solving time stays however very limited compared to the optimal MPC approaches. The relative deviation in infinite-horizon objective value δ_{V_∞} with respect to optimal MPC is however more pronounced for large N . This because the average number of PGM iterations n_{PGM} required to obtain the optimum increases for larger N .

6.1.3. Experimental results

When implementing the real-time PGM on the Arduino micro-controller, one cannot expect to obtain similar solving times as in simulation. The Arduino MEGA 2560’s ATmega2560 CPU has a clock frequency of only 16 MHz. Experimentally it was determined that the horizon length N should be reduced to 20 in order to be able to run the real-time PGM successfully at 50 Hz. Apart from the increased solving time, an increased horizon requires memory that easily exceeds the 8 kB of available RAM. These observations already

Figure 3: Influence of the horizon length N on the average solving time t_s , the share of the projection time t_{proj} in t_s and the relative deviation in infinite-horizon objective value δV_∞ with respect to optimal MPC. The average number of PGM iterations required to obtain the optimum is indicated by n_{PGM} .

suggest that in contrast to the real-time PGM, implementing optimal MPC on the Arduino is infeasible.

Figure 4 presents the step response obtained from an experiment. The generated input sequence and response follow the simulated trajectories. Possible deviations are mainly due to unmodeled nonlinear effects such as nonlinear friction in the motors' gearboxes. A video of the setup performing the step is attached to this paper.

6.1.4. Inverted pendulum control

The real-time PGM is employed to stabilize the pendulum in the inverted, unstable, position $\theta = \pi$ rad. After linearizing the system dynamics around this position, the model gets an extra unstable pole on top of the integrator. In theory, a proper terminal set should be included. However, with a short horizon length $N = 20$, and for disturbances occurring in practice, it is infeasible to satisfy a terminal state constraint which zeroes the unstable modes at the end of the horizon. In practice it is however possible to stabilize the pendulum while omitting the terminal set. The terminal control law $\kappa_f(\cdot)$ is chosen as a stabilizing static state feedback controller. To satisfy (3b), the terminal cost function $V_f(x) = \frac{1}{2}x^T Px$ is included where P is the solution of the Lyapunov equation, applied to the closed-loop dynamics following from $\kappa_f(\cdot)$.

The stage cost is chosen as

$$l(x, u) = \rho_1(\pi - \theta)^2 + \rho_2 z^2 + u^2. \quad (26)$$

As this cost, together with a relative large terminal cost, results in aggressive bang-bang input trajectories that impede stabilization, an extra term is added to the resulting QP objective which penalizes the input rate.

After fine-tuning the weights in (26), a stabilizing controller was retrieved. Figure 5 presents the input sequence u and the measured pendulum angle θ during a stabilization experiment. A video showing this result is attached to this paper.

6.2. Position control of a ball on a tilting table

6.2.1. Setup overview

In order to further illustrate the versatility of the presented approach, the real-time PGM is experimentally validated on the setup, presented in Figure 6, which consists of a table with a ball on top. The goal of

Figure 4: Experimental step response for the pendulum mass ($N = 20$). The black and gray lines indicate the experimental response and the simulated response respectively.

Figure 5: Pendulum angle θ and input u generated by the real-time PGM while stabilizing the inverted pendulum.

Figure 6: Schematic of a section of the tilting table that is responsible for the movement of the ball in the x -direction. The stepper motor drives a crank and rod mechanism which allows setting the table's inclination θ , which is the system's control input.

this experiment is controlling the position of the ball on the table by tilting the table. The table is hingedly suspended and is tilted by controlling the angle of two stepper motors. The top layer of the table consists of a resistive touchscreen that allows to measure the position (x, y) of the ball. The setup is equipped with an Arduino MEGA 2560 micro-controller which reads the touchscreen's resistance and the stepper motors' angle setpoints.

Tilting the table makes the ball accelerate due to gravity. When considering small tilting angles θ and ϕ , the acceleration of the ball is described by the following linearized equations:

$$\begin{aligned}\ddot{x} &= -g\frac{5}{7}\theta, \\ \ddot{y} &= -g\frac{5}{7}\phi,\end{aligned}\tag{27}$$

in which g represents the gravitational acceleration and the factor $5/7$ emerges as a manifestation of the ball's rotational inertia. Equations (27) are discretized using a central differentiation scheme with sample frequency 25 Hz. The discretized model is used to design a real-time PGM to control the position of the ball.

6.2.2. Experimental results

The goal of this experiment is to have the ball track a reference position, resulting in stage cost:

$$l(x, \theta) = \rho(r - x)^2 + \theta^2,$$

for the x -direction and a similar cost for the y -direction. Because aggressive motions of the table increase the risk of the ball losing contact with the table, a relatively small emphasis, $\rho = 1$ is put on the tracking, resulting in smooth control signals. Rather strict input constraints are imposed as an additional measure to avoid jerky control signals, $u \in [-3.5^\circ, 3.5^\circ]$, and a horizon $N = 20$ is chosen.

The resulting control law is implemented on the Arduino MEGA 2560 micro-controller for both the x - and y -direction, increasing the computational load by a factor of 2. Still, the control law can be executed at a rate of 25 Hz. As a reference, a step from $(0, -0.04)$ m to $(0, 0.04)$ m and back is chosen. The resulting experimental data is displayed in Figure 7, along with the simulation results. It shows that the control law is able to have the ball track the provided reference while respecting the bounds on the input. Moreover, the experimental data qualitatively matches the simulation. The largest deviation between simulation and experiment is found when the ball is decelerating. This is likely to be caused by the limited bandwidth of the servo motors which are modeled as being perfect. Nevertheless, the control law is able to cope with this modeling error, proving the robustness of the design.

Figure 7: Comparison of the simulated and measured response of the tilting table as a result of a reference step from $(0, -0.04)$ m to $(0, 0.04)$ m and back.

6.3. Software

In order to quickly design and simulate the real-time PGM, a small software package is provided¹. The software is written in Python and provides functionality to generate C++ code implementing a designed real-time PGM controller. This code is easily embedded on a micro-controller such as an Arduino.

7. Conclusion

This paper presents a real-time implementation of the proximal gradient method (PGM) for linear MPC. In each control update one step of the PGM is performed, while a next update is warm-started using the solution of the previous one. Given standard assumptions for linear MPC and standard assumptions on the PGM step size, the resulting closed-loop system is shown to be asymptotically stable. For linear systems with box input constraints, the method translates into simple arithmetics which are easily executed on resource-constrained hardware. This is demonstrated by experimental validations where the real-time PGM is implemented on an Arduino micro-controller to suppress the swinging of a pendulum and to control the position of a ball on top of a tilting table. This experiment and related simulations demonstrate the fast execution times of the real-time PGM at the expense of limited performance deterioration compared to optimal MPC.

References

- [1] J. B. Rawlings, D. Q. Mayne, Model predictive control: Theory and design, Nob Hill Pub., 2009.
- [2] A. Bemporad, M. Morari, V. Dua, E. N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, *Automatica* 38 (1) (2002) 3–20.
- [3] G. Takács, G. Batista, M. Gulán, B. Rohal'-Ilkiv, Embedded explicit model predictive vibration control, *Mechatronics* 36 (2016) 54–62.
- [4] H. Ferreau, C. Kirches, A. Potschka, H. Bock, M. Diehl, qpOASES: A parametric active-set algorithm for quadratic programming, *Mathematical Programming Computation* 6 (4) (2014) 327–363.

¹<https://github.com/meco-group/rtPGM>

- [5] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, C. N. Jones, Efficient interior point methods for multistage problems arising in receding horizon control, in: Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, IEEE, 2012, pp. 668–674.
- [6] G. Frison, H. B. Sørensen, B. Dammann, J. B. Jørgensen, High-performance small-scale solvers for linear model predictive control, in: Control Conference (ECC), 2014 European, IEEE, 2014, pp. 128–133.
- [7] J. V. Frasch, S. Sager, M. Diehl, A parallel quadratic programming method for dynamic optimization problems, Mathematical Programming Computation 7 (3) (2015) 289–329.
- [8] G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, C. N. Jones, et al., Operator splitting methods in control, Foundations and Trends® in Systems and Control 3 (3) (2016) 249–362.
- [9] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, M. Morari, Embedded online optimization for model predictive control at megahertz rates, IEEE Transactions on Automatic Control 59 (12) (2014) 3238–3251.
- [10] P. Giselsson, Improved fast dual gradient methods for embedded model predictive control, IFAC Proceedings Volumes 47 (3) (2014) 2303–2309.
- [11] S. Richter, M. Morari, C. N. Jones, Towards computational complexity certification for constrained MPC based on Lagrange relaxation and the fast gradient method, in: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on, IEEE, 2011, pp. 5223–5229.
- [12] S. Richter, C. N. Jones, M. Morari, Computational complexity certification for real-time MPC with input constraints based on the fast gradient method, IEEE Transactions on Automatic Control 57 (6) (2012) 1391–1403.
- [13] P. Patrinos, A. Bemporad, An accelerated dual gradient-projection algorithm for embedded linear model predictive control, IEEE Transactions on Automatic Control 59 (1) (2014) 18–33.
- [14] V. Nedelcu, I. Necoara, Q. Tran-Dinh, Computational complexity of inexact gradient augmented Lagrangian methods: application to constrained MPC, SIAM Journal on Control and Optimization 52 (5) (2014) 3109–3134.
- [15] H. Michalska, D. Q. Mayne, Robust receding horizon control of constrained nonlinear systems, IEEE Transactions on Automatic Control 38 (11) (1993) 1623–1633.
- [16] P. O. Scokaert, D. Q. Mayne, J. B. Rawlings, Suboptimal model predictive control (feasibility implies stability), IEEE Transactions on Automatic Control 44 (3) (1999) 648–654.
- [17] R. Van Parys, G. Pipeleers, Real-time proximal gradient method for linear MPC, in: Proceedings of the 2018 European Control Conference, 2018.
- [18] L. Vandenbergh, Optimization methods for large-scale systems, UCLA EE 236C Lecture Notes.
- [19] N. Parikh, S. Boyd, et al., Proximal algorithms, Foundations and Trends® in Optimization 1 (3) (2014) 127–239.
- [20] Y. Nesterov, Gradient methods for minimizing composite functions, Mathematical Programming 140 (1) (2013) 125–161.
- [21] M. N. Zeilinger, M. Morari, C. N. Jones, Soft constrained model predictive control with robust stability guarantees, IEEE Transactions on Automatic Control 59 (5) (2014) 1190–1202.
- [22] D. Limón, T. Alamo, F. Salas, E. F. Camacho, On the stability of constrained MPC without terminal constraint, IEEE transactions on Automatic Control 51 (5) (2006) 832–836.
- [23] C. G. Broyden, A class of methods for solving nonlinear simultaneous equations, Mathematics of computation 19 (92) (1965) 577–593.
- [24] A. Steinhauser, M. Verbandt, N. van Duijkeren, R. Van Parys, L. Jacobs, J. Swevers, G. Pipeleers, Low-cost carry-home mobile platforms for project-based evaluation of control theory, IFAC-PapersOnLine 50 (1) (2017) 9138–9143.