Fast Coordinate Descent Augmented Lagrangian Methods for linearized MPC

Liliana Maria Ghinea*, Daniela Lupu, Marian Barbu*, Ion Necoară****

**University "Dunărea de Jos" Galaţi, Romania*
(e-mail: liliana.ghinea@ugal.ro, marian.barbu@ugal.ro).
***University Politehnica Bucharest, Romania*
(e-mail:daniela.lupu@upb.ro, ion.necoara@upb.ro)

Abstract: This paper proposes Coordinate Descent Augmented Lagrangian based methods for solving linear and nonlinear Model Predictive Control (MPC) problems. The augmented Lagrangian relaxation to handle the equality constraints over the dynamics of the systems is used and dual accelerated gradient method based on inexact dual gradient information for solving the corresponding dual problem is applied. Then, three algorithms that solve the inner subproblem that arises from minimizing the Augmented Lagrangian are presented: the coordinate descent method, the accelerated proximal coordinate descent method and the restarted accelerated coordinate descent method. A detailed comparison between these methods is provided on several test cases.

Keywords: model predictive control, augmented lagrangian, accelerated gradient, coordinate descent.

1. INTRODUCTION

Model Predictive Control (MPC) is widely used in today's process industry due to its ability to handle constrained, multivariable control problems. The optimization methods used to solve MPC problems in real-time are divided into two separate categories: implicit and explicit MPC (Kvasnica, 2016). In implicit MPC, the control problem is solved in a sequence of steps, starting with the initial condition in each step. At run–time, these steps are combined to form a complete process. In explicit MPC, each section of the state space is divided into a set of sections, and the optimal control solution is performed in each section at design – time. At run – time, this process consists of a sequence of steps that start with the initial condition and end with the complete solution (Bemporad et al., 2002)

For linear systems, the MPC problem is usually posed as a convex quadratic problem (QP). Having the theoretical foundation of quadratic programming established in (Frank and Wolfe, 1956), many numerical techniques have been developed in the optimization literature by effectively exploiting the structure present in this class of problems. In general, we can classify algorithms that solve quadratic programs into three categories: active set methods, interior point methods, and (dual) first order methods.

Active set methods are based on the simple remark that solving a quadratic problem with equality constraints is equivalent to solving a linear system of equations. Therefore, at each iteration of such a method one needs to solve a linear system and update the active set (the term active set refers to the subset of constraints that are satisfied as equalities by the current estimate of the solution). Since the numerical complexity per iteration is cubic in the problem dimension, active set general purpose solvers work well for small- to medium-sized quadratic problems. A primal active set solver is implemented by Matlab's quadprog function.

Interior point methods eliminate the inequality constraints from the problem formulation, by adding a barrier term in the objective function to penalize the violation of the inequality constraints. The nonlinear convex problem that results from using a logarithmic barrier term is typically solved using the Newton method. Interior-point solvers are also the norm for small- to medium-scale QPs since the iteration complexity increases cubically with the dimension. For specific large-scale applications, however, structure-exploiting interior point solvers have also been developed. Large-scale structured quadratic programs have been the subject of a parallel interior point code developed in (Gondzio and Grothey, 2007).

First order methods compute a step toward the unconstrained problem's solution and then project this step onto the feasible set at each iteration, using just gradient information. Primal first order methods can be used to solve convex QPs with simple constraints. In this instance, a matrix-vector product represents the primary computational effort at each iteration.

Code simplicity is a demanding criterion for industrial MPC applications, particularly in applications that are safety-critical and require easy maintenance and verification of the code. In this regard, first-order methods (like projected gradient) are quite appealing due to their straightforward embedded implementations, as opposed to interior-point and active-set methods, which may necessitate more challenging linear algebra operations such as solving linear systems or effective matrix factorization operations to exploit the MPC structure, (Bemporad, 2018). On the other hand, Augmented Lagrangian methods (ALM) are particularly adequate for solving constrained optimization problems. They are similar to penalty methods in that they add a penalty term to the objective and also Lagrange multipliers, (Nedelcu et al., 2013). Although they are not identical, the augmented Lagrangian and the Lagrange multiplier approaches are linked, (Hestenes, 1969).

1.1 Contributions

For linear MPC we also use the augmented Lagrangian framework to move the constraints comings from dynamics into the cost and apply the accelerated gradient algorithm to solve the augmented dual problem, which we call the outer problem. However, for computing the augmented dual gradient one needs to minimize the augmented Lagrangian, we call this the inner problem. For minimizing the augmented Lagrangian we consider variants of coordinate descent. Coordinate descent methods have attracted a lot of attention lately, due to their use in machine learning, as observed for example in papers (Hsieh et al., 2008; Chang et al., 2008; Richtárik and Takáč, 2016; Necoara and Clipici, 2015). In this paper, we exploit the structure arising from linear MPC formulation when applying accelerated dual gradient in combination with coordinate descent.

More precisely, the MPC problem to be solved is first transformed into quadratic programming (QP) form, with equality and bound constraints. Therefore, we solve the QP problem using the accelerated gradient method for minimizing the augmented Lagrangian problem (outer problem), see (Kang et al., 2015). Because the Hessian matrix of the corresponding augmented dual function is almost in block diagonal form, we propose to solve the subproblem using various coordinate descent methods. Accelerated coordinate descent methods are not so well known in the control community, thus we show how efficient and fast these methods can be in the context of linear MPC. In particular, for solving the inner subproblem we use three methods: random coordinate descent method (Nesterov, 2012), accelerated proximal coordinate descent method that is relevant in the context of strongly convex problems and requires knowledge of the strong convexity constant of the objective function (Lin et al., 2014) and the restarted accelerated coordinate descent method developed in (Fercoq and Qu, 2020), which does not require knowledge of the strong convexity constant of the objective. For these three algorithms we consider two cases, the scalar case and the block case. The coordinate descent method for the block case has already been discussed in (Wu and Bemporad, 2022) and we propose accelerated variants, which turn out to be faster. These algorithms are applied on seven dynamical systems, five linear and two nonlinear, namely AFTI – 16 aircraft, helicopter system, mass – spring system, inverted pendulum system, four rank system, a wastewater treatment plant with activated sludge (WWTP) and an adiabatic continuous stirred tank reactor (CSTR).

The paper is organized as follows: in section 2, we present the MPC formulation, in section 3 we elaborate the augmented Lagrangian method that solves the outer problem and section 4 includes the three coordinate descent methods we chose to solve the inner problem. Section 5 presents the descriptions of the seven systems as well as tables containing number of iterations for the inner and outer problems and the CPU time.

1.2 Notation

In this paper, we use the following notations: $M \succ 0$ ($M \succcurlyeq 0$) denotes positive definiteness (semi – definiteness) of a square matrix M, V^T denotes the transpose of V, where V is a vector or a matrix, $\|v\|_2$ represents the Euclidean norm of vector v, $\|v\|_W^2 = v^T W v$ for a given $W \succ 0$, $[v_i]_{lb_i}^{ub_i}$ is the projection operator, $[v_i]_{lb_i}^{ub_i} = \begin{cases} ub_i, & v_i \geq ub_i \\ v_i, lb_i < v_i < ub_i \\ lb_i, & v_i \leq lb_i \end{cases}$.

2. MODEL PREDICTIVE CONTROL

In this section, we present the MPC problem for tracking and the reformulation as a QP problem following the steps in (Wu and Bemporad, 2022). The MPC problem is:

$$\begin{aligned} \min_{z_k, u_k} \ & \tfrac{1}{2}\textstyle\sum_{k=0}^{N-1} \|y_{k+1} - y_{k+1}^r\|_{W_y}^2 + \tfrac{1}{2}\|v_k - v_k^r\|_{W_v}^2 + \tfrac{1}{2}\|u_k\|_{W_u}^2 \\ \text{s.t.} \ & z_{k+1} = A_z z_k + B_v v_k, && \forall\, k = \overline{0, N-1} \\ & y_{k+1} = C_y z_{k+1}, && \forall\, k = \overline{0, N-1} \\ & v_k = v_{k-1} + u_k, && \forall\, k = \overline{0, N-1} \\ & lb_z \leq z_k \leq ub_z, && \forall\, k = \overline{1, N} \\ & lb_v \leq v_k \leq ub_v, && \forall\, k = \overline{0, N-1} \\ & lb_u \leq u_k \leq ub_u, && \forall\, k = \overline{0, N-1} \\ & z_0 = z, \quad v_{-1} = v, \end{aligned} \tag{1}$$

where $z_k \in \mathbb{R}^{n_z}$ represents the state vector, $v_k \in \mathbb{R}^{n_v}$ is the input vector, $y_k \in \mathbb{R}^{n_y}$ constitutes the output vector, $u_k = v_k - v_{k-1}$ is the vector of input increments, y_{k+1}^r and v_k^r are the output and input reference points, z and v denote the initial values for the state and input vectors. In order to simplify the notation, we reformulate the general problem (1) as follows:

$$\begin{aligned} \min_{s_k} \ & \frac{1}{2}\sum_{k=1}^{N} s_k^T (C^T W C)\, s_k - s_k^T (C^T W r_k) + \frac{1}{2} u_{k-1}^T W_u u_{k-1} \\ \text{s.t.} \ & s_{k+1} = A s_k + B u_k, && \forall\, k = \overline{0, N-1} \\ & lb_s \leq s_k \leq ub_s, && \forall\, k = \overline{1, N} \\ & lb_u \leq u_k \leq ub_u, && \forall\, k = \overline{0, N-1} \\ & s_0 = \begin{bmatrix} z_0 \\ v_{-1} \end{bmatrix}, \end{aligned}$$

where $s_k = \begin{bmatrix} z_k \\ v_{k-1} \end{bmatrix}$, $r_k = \begin{bmatrix} y_k^r \\ v_{k-1}^r \end{bmatrix}$, $A = \begin{bmatrix} A_z & B_v \\ 0 & I_{n_v} \end{bmatrix}$, $B = \begin{bmatrix} B_v \\ I_{n_v} \end{bmatrix}$, $C = \begin{bmatrix} C_y & 0 \\ 0 & I_{n_v} \end{bmatrix}$, $W = \begin{bmatrix} W_y & 0 \\ 0 & W_v \end{bmatrix}$, and lower bound lb_s is formed the same as the upper bound, i.e. $ub_s = \begin{bmatrix} ub_z \\ ub_v \end{bmatrix}$. Further, we denote by $x \in \mathbb{R}^{N \times n_x}$, with $n_x = n_z + 2n_v$, the decision variable that includes the state and input variables, as well as the vector of input increments over the entire prediction horizon N: $x = [u_0^T,\ s_1^T, u_1^T,\ s_1^T, \dots, u_{N-1}^T,\ s_N^T]^T \in \mathbb{R}^{N \times n_x}$. Then, the inequality constraints on state and input variables become:

$$lb \leq x \leq ub \ \Leftrightarrow \begin{cases} lb_s \leq s_k \leq ub_s, & \forall\, k = \overline{1, N} \\ lb_u \leq u_k \leq ub_u. & \forall\, k = \overline{0, N-1} \end{cases}.$$

Thus, at each step, the MPC problem can be written as the following quadratic problem (QP):

$$\begin{aligned} & \min_x \tfrac{1}{2} x^T H x + f^T x \\ & s.t.\ lb \leq x \leq ub \\ & \quad A_{eq} x = b_{eq}, \end{aligned} \tag{2}$$

where $H = H^T \succcurlyeq 0, H \in \mathbb{R}^{n\times n}$, $n = Nn_x, f \in \mathbb{R}^n, A_{eq} \in \mathbb{R}^{N(n_z+n_v)\times n}$ and $b_{eq} \in \mathbb{R}^{N(n_z+n_v)}$. The matrices have the following forms:

$$H = \begin{bmatrix} R & 0 & \cdots & 0 & 0 \\ 0 & Q & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & R & 0 \\ 0 & 0 & \cdots & 0 & Q \end{bmatrix}, \begin{cases} R = W_u \\ Q = C^T W\, C \end{cases}, \quad f = \begin{bmatrix} -C^T W\, r_1 \\ -C^T W\, r_2 \\ \vdots \\ -C^T W\, r_N \end{bmatrix} \tag{3}$$

$$A_{eq} = \begin{bmatrix} B & -I & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & A & B & -I & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & A & B & -I \end{bmatrix}, b_{eq} = \begin{bmatrix} -As_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Remark 1: Matrices $A_z, B_v, C_y, W_z, W_u, W_v$, as well as the upper and lower bounds on z, v, u may change at each controller iteration.

Nowadays, we are dealing with fast systems where the controller must be compiled rapidly and thus efficiency is highly required. In this context, we propose three algorithms: classical coordinate descent method, accelerated proximal coordinate descent method and restarted accelerated coordinate descent method. All three methods are used for finding solutions for the inner problem considered and are presented in the next section.

3. NUMERICAL SOLUTIONS FOR THE OUTER PROBLEM

In this section, we present an efficient algorithmic framework for solving (2), based on augmented Lagrangian (AL) and coordinate descent (CD). Taking into account the constraints of problem (2), primal methods cannot be applied since projection onto the feasible set is difficult. Hence, we consider using dual methods and one of the most efficient is the dual fast augmented Lagrangian method. Thus, let us give the expression for the bound – constrained Lagrangian function $\mathcal{L}: \mathcal{X} \times \mathbb{R}^n \to \mathbb{R}$, with $\mathcal{X} = \{ lb \le x \le ub \}$:

$$\mathcal{L}(x,\lambda) = \tfrac{1}{2}x^T Hx + f^T x + \lambda^{\mathrm{T}}(A_{eq}x - b_{eq}), \tag{4}$$

where $\lambda \in \mathbb{R}^{N(n_z+n_v)}$. is the vector of Lagrange multipliers associated with the equality constraints in QP (2). Thus, the dual problem of (2) is:

$$\max_{\lambda} d(\lambda) = \max_{\lambda} \min_{x\in\mathcal{X}} \mathcal{L}(x,\lambda) \tag{5}$$

The downside is that, in general, the function $d(\lambda)$ is not differentiable (Bertsekas, 1997) and thus any subgradient method employed for solving (5) would have a slow convergence rate. Therefore, we use the augmented Lagrangian function

$$\mathcal{L}_\rho(x,\lambda) = \tfrac{1}{2}x^T Hx + f^T x + \lambda^{\mathrm{T}}\big(A_{eq}x - b_{eq}\big) + \tfrac{\rho}{2}\big\|A_{eq}x - b_{eq}\big\|^2 \tag{6}$$

instead, where $\rho > 0$ is a penalty parameter. The dual augmented Lagrangian problem becomes:

$$\max_{\lambda} d_\rho(\lambda) = \max_{\lambda} \min_{x\in\mathcal{X},} \mathcal{L}_\rho(x,\lambda), \tag{7}$$

where $d_\rho(\lambda)$ is differentiable function if $H + \rho A_{eq}^T A_{eq} \succ 0$. The dual problem (5) and the augmented dual problem (7) share the same optimal solution (Bertsekas, 1982). The objective function of the dual problem (7) is concave and has Lipschitz continuous gradient with Lipschitz constant $\frac{1}{\rho}$ (Bertsekas, 1997; Nesterov, 2005):

$$\nabla d_\rho(\lambda) = A_{eq}x^*(\lambda) - b_{eq}, \tag{8}$$

where $x^*(\lambda)$ is the notation used for the optimal solution for the inner problem,

$$\min_{x\in\mathcal{X}} \mathcal{L}_\rho(x,\lambda) \tag{9}$$

for a given λ. Note that the inner problem can be written in the following form:

$$\mathcal{L}_\rho(x,\lambda) = \tfrac{1}{2}x^T\mathcal{H}x + \mathcal{f}(\lambda)^T x, \tag{10}$$

where $\mathcal{H} = H + \rho A_{eq}^T A_{eq}$ and $\mathcal{f}(\lambda) = f + A_{eq}^T\lambda - \rho A_{eq}^T b_{eq}$ (see (6)). Notice that matrix $\mathcal{H}$ has a block sparse structure of the following form (Wu and Bemporad, 2022)

$$\mathcal{H} = \begin{bmatrix} \psi_1 & \psi_2 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \psi_2^T & \psi_3 & \psi_4 & \psi_5 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \psi_4^T & \psi_1 & \psi_2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \psi_5^T & \psi_2^T & \psi_3 & \psi_4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \psi_3 & \psi_4 & \psi_5 \\ 0 & 0 & 0 & 0 & 0 & \cdots & \psi_4^T & \psi_1 & \psi_2 \\ 0 & 0 & 0 & 0 & 0 & \cdots & \psi_5^T & \psi_2^T & \psi_6 \end{bmatrix}, \tag{11}$$

with:

$$\psi_1 = R + \rho B^T B\,,\ \psi_2 = -\rho B^T\,,\ \psi_3 = Q + \rho I + \rho A^T A,$$

$$\psi_4 = \rho A^T B\,,\ \psi_5 = -\rho A^T\,,\ \psi_6 = Q + \rho I.$$

The augmented Lagrangian algorithm has the following iterations, Bertsekas (1982):

$$x_{k+1} = \operatorname*{argmin}_{x\in\mathcal{X}} \mathcal{L}_\rho(x,\lambda_k) \tag{12}$$

$$\lambda_{k+1} = \lambda_k + (A_{eq}x_{k+1} - b_{eq}). \tag{13}$$

As demonstrated in (Bertsekas, 1982), the convergence of the augmented Lagrangian algorithm can be ensured for a large range of values for the penalty parameter ρ. More specifically, the larger the value of ρ, the faster the algorithm is to converge, but the harder subproblem (12) is to solve. This happens because of the larger condition number of the Hessian matrix for subproblem (12). The convergence rate of the augmented Lagrangian algorithm is $O\left(\frac{1}{t}\right)$, as shown in (He and Yuan, 2010). In order to boost the speed of the algorithm, we choose to work with the accelerated version of the augmented Lagrangian method, (Kang et al., 2015). The convergence rate of the accelerated augmented Lagrangian method is $O\left(\frac{1}{t^2}\right)$ for linearly constrained convex programs, by using Nesterov's accelerated gradient method. The accelerated augmented Lagrangian method is presented below:

AALM (Accelerated Augmented Lagrangian Method) (Kang et al., 2015)

Input: $x_0 \in \mathcal{X}$, $\lambda_0 \in \mathbb{R}^{N(n_z+n_v)}$ initial guess, $\rho > 0$ penalty parameter, T maximum number of iterations for the outer problem, ε_{out} outer accuracy

1. **set** $\tau_0 = 1, \nu_0 = \lambda_0$
2. **for** $t = 0:T-1$ **do**
   2.1. $x_t = \underset{x}{\operatorname{argmin}}\, \mathcal{L}_\rho(x, \nu_t)$
   2.2. $\lambda_{t+1} = \nu_t + \rho(A_{eq}x_t - b_{eq})$
   2.3. **if** $\|\lambda_{t+1} - \nu_t\|_2^2 \leq \varepsilon_{out}$ **break**
   2.4. $\tau_{t+1} = \frac{1+\sqrt{1+4\tau_t^2}}{2}$
   2.5. $\nu_{t+1} = \lambda_{t+1} + \frac{\tau_t - 1}{\tau_{t+1}}(\lambda_{t+1} - \lambda_t)$
3. **end**

Note that in step 2.1 we need to solve the strongly convex inner subproblem (see (10)). Since d_ρ is concave and smooth, the convergence rate for the previous algorithm is $O\left(\frac{1}{t^2}\right)$.

4. NUMERICAL SOLUTIONS FOR THE INNER SUBPROBLEM

Note that the inner subproblem has the following form:

$$\min_{x\in\mathcal{X}} \mathcal{L}_\rho(x, \nu) := \frac{1}{2}x^T\mathcal{H}x + \not{h}(\nu)^T x\,, \tag{14}$$

where $\mathcal{H} \succ 0$. Hence, the cost is strongly convex and the strong convexity parameter is the minimum eigenvalue of $\mathcal{H}$, $\mu = \lambda_{min}(\mathcal{H})$. There are problems where finding the strong convexity parameter is difficult, as well as cases when the value can be easily computed. Thus, we propose three solutions that cover these cases: coordinate descent method, accelerated proximal coordinate descent method and restarted accelerated coordinate descent method. We adapt these three algorithms for two cases: the scalar case, where we minimize problem (14) along one randomly chosen coordinate direction at each iteration, while keeping the other coordinates fixed and the block case, where we update block components of the states and controls of the system, in cyclic reverse order.

In the scalar case, the whole matrix $\mathcal{H}$ is required as an input variable, whereas for the block case we need form (11) of this matrix. More specifically, we use $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6$. For the block case, recall notation $s_k = \begin{bmatrix} z_k \\ v_{k-1} \end{bmatrix}$, thus x_k becomes $x_k = \begin{bmatrix} u_{k-1} \\ s_k \end{bmatrix}$ and therefore calculate $s_N, u_{N-1}, \dots, s_1, u_0$. Keep in mind that the Lipschitz constants are $L_N^s = ||\psi_6||$, $L_{N-1}^s = L_{N-2}^s = \cdots = L_1^s = ||\psi_3||$ and $L_{N-1}^u = L_{N-2}^u = \dots L_0^u = ||\psi_1||$. For this case, we use the reverse cyclic rule in order to make use of the fact that shifted previous optimal solution is used as a warm start.

4.1 Coordinate descent method

The coordinate descent method (Nesterov, 2012), is presented in both scalar and block form and it is adapted to problem (14). Note that for the CD method, the value for the strong convexity parameter is not required.

CD (Coordinate Descent Method for the scalar case)

Input: $x^0 \in \mathbb{R}^{n_x}$, $\nu \in \mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, ε_{in} inner accuracy, M number of points to consider for stopping criteria.

1. **for** $\ell = 0:T_{inner}-1$
   1.1. $i = randi(n_x, 1)$
   1.2. $x_i^{\ell+1} = \left[x_i^\ell - \frac{1}{\mathcal{H}_{i,i}}\left(\mathcal{H}_{i,\cdot}x^\ell + \not{h}_i(\nu)^\ell\right)\right]_{lb^\ell}^{ub^\ell}$
   1.3. $x_j^{\ell+1} = x_j^\ell, j \neq i$
   1.4. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$ **break**
2. **end**

In the step 1.1 of the CD algorithm, we generate uniformly random an integer for accessing an index in variable x. In the following step, we compute a gradient descent update and project onto the interval constrain, which can be done easily (see subsection 1.2.). We start checking the stopping criteria over the last M iterations by looking at the maximum distance between two consecutive calculated points. This strategy for the stopping criteria is applied to all following algorithms. Further, the cyclic block case version of this algorithm is presented:

CD BLOCK (Coordinate Descent Method for block case)

Input: $x^0 \in \mathbb{R}^{n_x}$, $\mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6$ (see (11)), ε_{in} inner accuracy, M.

1. **for** $\ell = 0:T_{inner}-1$
   1.1. $s_N^{\ell+1} = \left[z_N^\ell - \frac{1}{L_N^z}\left(\psi_5^T s_{N-1}^\ell + \psi_2^T u_{N-1}^\ell + \psi_6 s_N^\ell + \not{h}_N(\nu)\right)\right]_{lb_N}^{ub_N}$
   1.2. $u_{N-1}^{\ell+1} = \left[u_{N-1}^\ell - \frac{1}{L_{N-1}^u}\left(\psi_4^T s_{N-1}^\ell + \psi_1 u_{N-1}^\ell + \psi_2 s_N^\ell + \not{h}_{N-1}(\nu)\right)\right]_{lb_{N-1}}^{ub_{N-1}}$
   1.3. **for** $i = N-1:2$ **do**
      1.3.1. $s_i^{\ell+1} = \left[s_i^\ell - \frac{1}{L_i^s}\left(\psi_5^T s_{i-1}^\ell + \psi_2^T u_{i-1}^\ell + \psi_3 s_i^\ell + \psi_4 u_i^\ell + \psi_5 s_{i+1}^\ell + \not{h}_i(\nu)\right)\right]_{lb_i}^{ub_i}$
      1.3.2. $u_{i-1}^{\ell+1} = \left[u_{i-1}^\ell - \frac{1}{L_{i-1}^u}\left(\psi_4^T s_{i-2}^\ell + \psi_1 u_{i-2}^\ell + \psi_2 s_{i-1}^\ell + \not{h}_{i-1}(\nu)\right)\right]_{lb_{i-1}}^{ub_{i-1}}$
      1.3.3. **end**

1.4. $s_1^{\ell+1} = \left[s_1^{\ell} - \frac{1}{L_1^s}\left(\psi_2^T u_0^{\ell} + \psi_3 s_1^{\ell} + \psi_4 u_1^{\ell} + \psi_5 s_2^{\ell} + \phi_1(v)\right)\right]_{lb_1}^{ub_1}$

1.5. $u_0^{\ell+1} = \left[u_0^{\ell} - \frac{1}{L_0^u}\left(\psi_1 u_0^{\ell} + \psi_2 s_1^{\ell} + \phi_0(v)\right)\right]_{lb_0}^{ub_0}$

1.6. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$

1.6.1. **break**

2. **end**

Note that the algorithm converges linearly in both cases, since the QP subproblem is strongly convex and smooth.

4.2 Accelerated proximal coordinate gradient method

The accelerated proximal coordinate gradient (APCG) method, derived in paper (Lin et al., 2014), is presented in both scalar and block form and it is adapted to problem (14). Note that for applying the APCG method, we need to know the value for the strong convexity parameter μ.

Assumption 1 (Lin et al. (2014)): There exists $\mu > 0$ such that $\forall\, x, w \in \mathbb{R}^{N_x}$ the following holds:

$$\mathcal{L}_\rho(w) \geq \mathcal{L}_\rho(x) + \langle \nabla \mathcal{L}_\rho(x), w - x\rangle + \frac{\mu}{2}\|w - x\|_L^2.$$

Taking into consideration all of the above, the APCG algorithm (random scalar variant) is presented in the following form:

APCG (Accelerated Proximal Coordinate Gradient)

Input: $x^0 \in \mathbb{R}^{n_x}$, $v \in \mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, strong convexity parameter $\mu > 0$, ε_{in} inner accuracy, M

1. **set** $\phi^0 = x^0$, $w^0 = x^0$, $\alpha = \frac{\sqrt{\mu}}{n_x}$
2. **for** $\ell = 0{:}\,T_{inner} - 1$

2.1. $w^{\ell+1} = \frac{x^{\ell} + \alpha\phi^{\ell}}{1+\alpha}$

2.2. $\phi^{\ell+1} = (1-\alpha)\phi^{\ell} + \alpha w^{\ell+1}$

2.3. $i = randi(n_x, 1)$

2.3.1. $L_i = \left|\mathcal{H}_{i,i}\right|$

2.3.2. $\phi_i^{\ell+1} = \left[\phi_i^{\ell} - \frac{1}{n_x \alpha L_i}\left(\mathcal{H}_{i,\cdot} w^{\ell+1} + \phi_i(v)^{\ell}\right)\right]_{lb^{\ell}}^{ub^{\ell}}$

2.3.3. $x^{\ell+1} = w^{\ell+1} + n_x\alpha\left(\phi^{\ell+1} - \phi^{\ell}\right) + n_x\alpha^2(w^{\ell} - y^{\ell+1})$

2.4. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$

2.4.1. **break**

3. **end**

The following is the version of APCG for the cyclic block case:

APCG BLOCK (Accelerated Proximal Coordinate Gradient for block case)

Input: $x^0 \in \mathbb{R}^{n_x}$, $v \in \mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6$ (see (11)), ε_{in} inner accuracy, M.

1. **set** $\phi^0 = x^0$, $w^0 = x^0$, $\alpha = \frac{\sqrt{\mu}}{n_x}$, $\eta = 2N$
2. **for** $\ell = 0{:}\,T_{inner} - 1$

2.1. $w^{\ell+1} = \frac{x^{\ell} + \alpha\phi^{\ell}}{1+\alpha}$

2.2. $\phi^{\ell+1} = (1-\alpha)\phi^{\ell} + \alpha w^{\ell+1}$

2.3. $s_N{}^{\ell+1} = \left[s_N^{\ell} - \frac{1}{\eta\alpha L_N^s}(\psi_5^T s_{N-1}^{\ell} + \psi_2^T u_{N-1}^{\ell} + \psi_6 s_N^{\ell} + \phi_N(v))\right]_{lb_N}^{ub_N}$

2.4. $u_{N-1}{}^{\ell+1} = \left[u_{N-1}{}^{\ell} - \frac{1}{\eta\alpha L_{N-1}^u}(\psi_4^T s_{N-1}^{\ell} + \psi_1 u_{N-1}^{\ell} + \psi_2 s_N^{\ell} + \phi_{N-1}(v))\right]_{lb_{N-1}}^{ub_{N-1}}$

2.5. **for** $i = N-1{:}\,2$ **do**

2.5.1. $s_i^{\ell+1} = \left[s_i^{\ell} - \frac{1}{\eta\alpha L_i^s}(\psi_5^T s_{i-1}^{\ell} + \psi_2^T u_{i-1}^{\ell} + \psi_3 s_i^{\ell} + \psi_4 u_i^{\ell} + \psi_5 s_{i+1}^{\ell} + \phi_i(v))\right]_{lb_i}^{ub_i}$

2.5.2. $u_{i-1}^{\ell+1} = \left[u_{i-1}^{\ell} - \frac{1}{\eta\alpha L_{i-1}^u}(\psi_4^T s_{i-2}^{\ell} + \psi_1 u_{i-2}^{\ell} + \psi_2 s_{i-1}^{\ell} + \phi_{i-1}(v))\right]_{lb_{i-1}}^{ub_{i-1}}$

2.5.3. **end**

2.6. $s_1{}^{\ell+1} = \left[s_1^{\ell} - \frac{1}{\eta\alpha L_1^s}(\psi_2^T u_0^{\ell} + \psi_3 s_1^{\ell} + \psi_4 u_1^{\ell} + \psi_5 s_2^{\ell} + \phi_1(v))\right]_{lb_1}^{ub_1}$

2.7. $u_0{}^{\ell+1} = \left[u_0{}^{\ell} - \frac{1}{\eta\alpha L_0^u}(\psi_1 u_0^{\ell} + \psi_2 s_1^{\ell} + \phi_0(v))\right]_{lb_0}^{ub_0}$

2.8. $\phi^{\ell+1} = \begin{bmatrix} s_0{}^{\ell+1} \\ u_0{}^{\ell+1} \\ s_1{}^{\ell+1} \\ \vdots \\ u_{N-1}{}^{\ell+1} \\ s_N{}^{\ell+1} \end{bmatrix}$

2.9. $x^{\ell+1} = w^{\ell+1} + n_x\alpha\left(\phi^{\ell+1} - \phi^{\ell}\right) + n_x\alpha^2(\phi^{\ell} - w^{\ell+1})$

2.10. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$

2.10.1. **break**

3. **end**

Note that the APCG method for minimizing strongly convex functions achieves faster linear convergence rates than coordinate gradient methods.

4.3 Restarted accelerated coordinate descent method

(Fercoq and Qu, 2020) considers a restarting strategy for the accelerated coordinate descent method. We present it for both the scalar and block case and adapt it to problem (14). Note that for applying the APCG method, we do need to know the value for the strong convexity parameter μ, while in the restarted version below we do not need μ.

APPROX (Accelerated Parallel and Proximal Coordinate Descent Method)

Input: $x^0 \in \mathbb{R}^{n_x}$, $v \in \mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, ε_{in} inner accuracy, M.

1. **set** $m^0 = x^0, \theta^0 = \frac{1}{n_x}$
2. **for** $\ell = 0 : T_{inner} - 1$
   2.1. $y^{\ell+1} = (1-\theta^\ell)x^\ell + \theta^\ell m^\ell$
   2.2. $i = randi(n_x, 1)$
      2.2.1. $L_i = |\mathcal{H}_{i,i}|$
      2.2.2. ${m_i}^{\ell+1} = \left[{m_i}^\ell - \frac{1}{\theta^\ell n_x L_i}(\mathcal{H}_{i,\cdot} y^{\ell+1} + \mathcal{f}_i(v)^\ell)\right]_{lb^\ell}^{ub^\ell}$
      2.2.3. $x^{\ell+1} = y^{\ell+1} + n_x\theta^\ell(m^{\ell+1} - m^\ell)$
   2.3. $\theta^{\ell+1} = \frac{\sqrt{(\theta^\ell)^4 + 4(\theta^\ell)^2} - (\theta^\ell)^2}{2}$
   2.4. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$
      2.4.1. **break**
3. **end**

The following is the version of APPROX for the block case:

APPROX BLOCK (Accelerated Proximal Coordinate Gradient Method for the block case)

Input: $x^0 \in \mathbb{R}^{n_x}$, $v \in \mathbb{R}^{N(n_z+n_v)}$, T_{inner} number of iterations, $\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6$ (see (11)), ε_{in} inner accuracy, M.

1. **set** $m^0 = x^0, \theta^0 = \frac{1}{n_x}, \eta = 2N$
2. **for** $\ell = 0 : T_{inner} - 1$
   2.1. $y^{\ell+1} = (1-\theta^\ell)x^\ell + \theta^\ell m^\ell$
   2.2. ${s_N}^{\ell+1} = \left[s_N^\ell - \frac{1}{\theta^\ell \eta L_N^s}(\psi_5^T s_{N-1}^\ell + \psi_2^T u_{N-1}^\ell + \psi_6 s_N^\ell + \mathcal{f}_N(v))\right]_{lb_N}^{ub_N}$
   2.3. ${u_{N-1}}^{\ell+1} = \left[{u_{N-1}}^\ell - \frac{1}{\theta^\ell \eta L_{N-1}^u}(\psi_4^T s_{N-1}^\ell + \psi_1 u_{N-1}^\ell + \psi_2 s_N^\ell + \mathcal{f}_{N-1}(v))\right]_{lb_{N-1}}^{ub_{N-1}}$
   2.4. **for** $i = N-1 : 2$ **do**
      2.4.1. $s_i^{\ell+1} = \left[s_i^\ell - \frac{1}{\theta^\ell \eta L_i^s}(\psi_5^T s_{i-1}^\ell + \psi_2^T u_{i-1}^\ell + \psi_3 s_i^\ell + \psi_4 u_i^\ell + \psi_5 s_{i+1}^\ell + \mathcal{f}_i(v))\right]_{lb_i}^{ub_i}$
      2.4.2. $u_{i-1}^{\ell+1} = \left[u_{i-1}^\ell - \frac{1}{\theta^\ell \eta L_{i-1}^u}(\psi_4^T s_{i-2}^\ell + \psi_1 u_{i-2}^\ell + \psi_2 s_{i-1}^\ell + \mathcal{f}_{i-1}(v))\right]_{lb_{i-1}}^{ub_{i-1}}$
      2.4.3. **end**
   2.5. ${s_1}^{\ell+1} = \left[s_1^\ell - \frac{1}{\theta^\ell \eta L_1^s}(\psi_2^T u_0^\ell + \psi_3 s_1^\ell + \psi_4 u_1^\ell + \psi_5 s_2^\ell + \mathcal{f}_1(v))\right]_{lb_1}^{ub_1}$
   2.6. ${u_0}^{\ell+1} = \left[{u_0}^\ell - \frac{1}{\theta^\ell \eta L_0^u}(\psi_1 u_0^\ell + \psi_2 s_1^\ell + \mathcal{f}_0(v))\right]_{lb_0}^{ub_0}$
   2.7. $m^{\ell+1} = \begin{bmatrix} {s_0}^{\ell+1} \\ {u_0}^{\ell+1} \\ {s_1}^{\ell+1} \\ \vdots \\ {u_{N-1}}^{\ell+1} \\ {s_N}^{\ell+1} \end{bmatrix}$
   2.8. $x^{\ell+1} = y^{\ell+1} + \eta\theta^\ell(m^{\ell+1} - m^\ell)$
   2.9. $\theta^{\ell+1} = \frac{\sqrt{(\theta^\ell)^4 + 4(\theta^\ell)^2} - (\theta^\ell)^2}{2}$
   2.10. **if** $\max_{k=0:M-1}\left(\left\|x^{\ell-k+1} - x^{\ell-k}\right\|^2\right) > \varepsilon_{in}$
      2.10.1. **break**
3. **end**

In order to ensure linear convergence, a restarting scheme which adaptively approximates the strong convexity parameter must be applied:

R - APPROX (APPROX with restart)

1. **choose** $x_0 \in \mathbb{R}^{n_x}$
2. **set** $\hat{x}_0 = x_0$
3. **choose** restart periods $\{T_{inner}^0, T_{inner}^1, \dots, T_{inner}^r, \dots\} \subseteq \mathbb{N}$
4. **for** $r \geq 0$ **do**
   4.1. $\hat{x}_{r+1} = APPROX((14), \hat{x}_r, T_{inner}^r)$
5. **end**

5. SIMULATIONS

In this section we test the performance of accelerated gradient dual augmented Lagrangian method combined with the three coordinate descent methods presented before on five linear systems and two nonlinear ones and compare the results of the algorithms, both in terms of iterations and CPU time. All algorithms are written in Matlab R2018b, and the simulations have been performed on a 64-bit operating system PC with Processor Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz 2.21 GHz, and 16.0 GB RAM. We begin by presenting the five linear systems and two nonlinear ones and the simulation conditions.

5.1 Linear systems

AFTI - 16 system

The open-loop unstable linearized AFTI-16 aircraft model presented in (Kapasouris et al., 1988) is represented by the continuous-time system matrices:

$$A_c = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.174 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.00018 & 43.2541 & -0.86939 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} -2.516 & -13.136 \\ -0.1689 & -0.2514 \\ -17.251 & -1.5766 \\ 0 & 0 \end{bmatrix},\ C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The input constraints are $|v_i| \leq 25,\ i = \overline{1,2}$, the constraints over the states are $-100 \leq z_2 \leq 100, -0.5 \leq z_4 \leq 0.5$, the output variable is $y = [z_2\ z_4]^T$ and the starting point is $z_0 = [0\ 0\ 0\ 0]^{\mathrm{T}}$. The system is discretized using the Euler method. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = I_{n_v}$, $W_y = I_{n_y}$.

Helicopter system

The laboratory model helicopter (Quanser 3-DOF Helicopter) presented in (Tøndel and Johansen, 2002) is represented by the discrete-time system matrices:

$$A_d = \begin{bmatrix} 1 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 0 \\ 0.0001 & -0.0001 \\ 0.0019 & 0.0019 \\ 0.0132 & -0.0132 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},\ C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The input constraints are $-1 \leq v_i \leq 3,\ i = \overline{1,2}$, whereas the constraints over the states are $-0.44 \leq z_3 \leq 0.44$ and $-0.6 \leq z_4 \leq 0.6$, the output variable is $y = [z_3\ z_4]^T$ and the starting point is $z_0 = [0.5\ 0.5\ 0\ 0\ 0\ 0]^T$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = 0.1I_{n_v}$, $W_y = 10I_{n_y}$.

Mass – spring system

The mass – spring system presented in (Longo et al., 2013) is represented by the discrete-time system matrices:

$$A_d = \begin{bmatrix} 0.(9) & 0 & 0 & -0.02 & 0.01 & 0 \\ 0 & 0.(9) & 0 & 0.01 & -0.02 & 0.01 \\ 0 & 0 & 0.(9) & 0 & 0.01 & -0.02 \\ 0.01 & 0 & 0 & 0.(9) & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0.(9) & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0.(9) \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 & 0 \\ 0.01 & 0 \\ 0 & -0.0001 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0.01 \end{bmatrix},\ C = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

The input constraints are $-0.5 \leq v_i \leq 0.5,\ i = \overline{1,2}$, whereas the constraints over the states are $-3.5 \leq z_i \leq 3.5,\ i = \overline{1,6}$, the output variable is $y = [z_3\ z_4]^T$ and the starting point is $z_0 = [0\ 0\ 3.5\ 3.5\ 0\ 0]^{\mathrm{T}}$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = 0.1I_{n_v}$, $W_y = I_{n_y}$.

Inverted pendulum system

The inverted pendulum system presented in (Maeder et al., 2007) is represented by the discrete-time system matrices:

$$A_d = \begin{bmatrix} 1.001 & -0.05 & -0.001 \\ -0.05 & 1.003 & 0.05 \\ -0.001 & 0.05 & 1.001 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0 \\ 0.001 \\ 0.05 \end{bmatrix},\ C = [0 \quad 1 \quad 0]$$

The input constraint is $-1.25 \leq v \leq 1.25$, the output variable is $y = z_2$ and the starting point is $z_0 = [0.6\ 0.6\ 0.6]^T$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = 0.1I_{n_v}$, $W_y = 10I_{n_y}$.

Four – tank system

The four – tank system presented in (Annamalai, 2015) is represented by the discrete-time system matrices:

$$A_d = \begin{bmatrix} -0.0158 & 0 & 0.0256 & 0 \\ 0 & -0.0109 & 0 & 0.0178 \\ 0 & 0 & -0.0256 & 0 \\ 0 & 0 & 0 & -0.0178 \end{bmatrix}$$

$$B_d = \begin{bmatrix} 0.0482 & 0 \\ 0 & 0.035 \\ 0 & 0.0775 \\ 0.0599 & 0 \end{bmatrix},\ C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The input constraints are $0 \leq v_i \leq 15,\ i = \overline{1,2}$, the constraints over the states are $0 \leq z_i \leq 15,\ i = \overline{1,4}$, the output variable is $y = [z_2\ z_4]^T$ and the starting point is $z_0 = [12.6\ 13\ 4.8\ 4.9]^{\mathrm{T}}$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = I_{n_v}$, $W_y = I_{n_y}$.

5.2 Nonlinear systems

In order to apply our algorithms on nonlinear systems, a few steps are required. Firstly, we denote by F_c the continuous function that describes the system and by F_d the discrete one:

$$\dot{z} = F_c(z, v)$$

$$z_{t+1} = F_d(z_t, v_t).$$

obtained by applying discretization with the Euler method, i.e.:

$$z_{t+1} = z_t + dNF_c(z_t, v_t),$$

where by dN we denote the sampling time. The next step is linearization:

$$F_d(z_t, v_t) \approx F_d(z_0, v_0) + \frac{\partial F_d}{\partial (z_t, v_t)}(z_0, v_0)\begin{bmatrix} z_t - z_0 \\ v_t - v_0 \end{bmatrix}$$

$$= F_d(z_0, v_0) + \begin{bmatrix} \frac{\partial F_d}{\partial z_t}(z_0, v_0) & \frac{\partial F_d}{\partial v_t}(z_0, v_0) \end{bmatrix}\begin{bmatrix} z_t - z_0 \\ v_t - v_0 \end{bmatrix}$$

$$= F_d(z_0, v_0) + \frac{\partial F_d}{\partial z_t}(z_0, v_0)(z_t - z_0) + \frac{\partial F_d}{\partial z_t}(z_0, v_0)(v_t - v_0).$$

Considering all of the above, we obtain the matrices as follows:

$$A_d = \frac{\partial F_d}{\partial z_i}(z_0, v_0) = I + dN\frac{\partial F_c}{\partial z_i}(z_0, v_0)$$

$$B_d = \frac{\partial F_d}{\partial v_i}(z_0, v_0) = dN\frac{\partial F_c}{\partial v_i}(z_0, v_0).$$

WWTP system

We consider a wastewater treatment system with four states, presented in (Caraman et al., 2006). The control variable is $v = W\ [mg\ DO/l]$, the output variable is $y = S\ [mg/l]$, the states are $z = \{X, S, DO, X_r\}$ and the differential equations that describe the system are:

$$\frac{\partial X}{\partial t} = \mu(t)X(t) - D(1 + r)X(t) + rDX_r(t)$$

$$\frac{\partial S}{\partial t} = -\frac{\mu(t)}{Y}X(t) - D(1 + r)S(t) + DS_{in}$$

$$\frac{\partial DO}{\partial t} = -\frac{K_0\mu(t)}{Y}X(t) - D(1 + r)DO(t) + \alpha W(t)\big(DO_{max} - DO(t)\big) + DDO_{in}$$

$$\frac{\partial X_r}{\partial t} = D(1 + r)X(t) - D(\beta + r)X_r(t)$$

with $\mu(t) = \mu_{max}\frac{S(t)}{K_S + S(t)}\frac{DO(t)}{K_{DO} + DO(t)}$ and the fixed parameters $\mu_{max} = 0.15, K_s = 100, K_{DO} = 2, DO_{max} = 10,$

$D = 0.1, DO_{in} = 0.5, S_{in} = 200, Y = 0.65, K_0 = 0.5,$

$\alpha = 0.018, \beta = 0.2, r = 0.6$. For this system, we apply discretization with Euler and linearization as described above in order to obtain the matrices required for the algorithms. The input constraints are $0 \le v \le 40$, the constraints over the states are $0 \le z_1 \le 200, 0 \le z_2 \le 40, 1 \le z_3 \le 3, 0 \le z_4 \le 400$, the output variable is $y = z_2$ and the starting point is $z_0 = [200\ 90\ 2\ 320]^T$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = 0.1I_{n_v}$, $W_y = 10I_{n_y}$.

CSTR system

The adiabatic continuous stirred tank reactor (CSTR) is a common chemical nonlinear system in the process industry, and it is presented in (Bequette, 1998). The CSTR system is modeled by using basic mass balance and energy conservation principles. A linearized model of the CSTR system is:

$$A_c = \begin{bmatrix} -5 & -0.3427 \\ 47.68 & 2.785 \end{bmatrix},\ B_c = \begin{bmatrix} 0 \\ 0.3 \end{bmatrix},\ C = [1 \quad 0]\ .$$

The input constraints are $273 \le v \le 322$, the constraints over the states are $0 \le z_1 \le 10, 310 \le z_2 \le 390$, the output variable is $y = z_1$ and the starting point is $z_0 = [8.5698\ 311.2639]^T$. As a performance metric we use a quadratic cost with the matrices Q and R defined in (3), thus $W_v = 0_{n_v}$, $W_u = 0.1I_{n_v}$, $W_y = I_{n_y}$.

For all these seven systems, the sampling time is $dN = 0.05$ and the prediction horizon is $N = 5$.

The stopping criteria are as follows: for the inner problem, the algorithm stops when the number of iterations reaches a certain number (200000 for CD, 50000 for APCG, 5000 for R – APPROX, 50000 for CD BLOCK, APCG BLOCK and R – APPROX BLOCK) or $\varepsilon_{in} = 10^{-7}$; for the outer problem, the algorithm stops when the number of iterations reaches 5000 $\varepsilon_{out} = 10^{-4}$.

The following figures represent the graphical comparisons for the AFTI - 16 system and the WWTP system. We compare the three algorithms presented in the previous section, both on the scalar case and the block case.

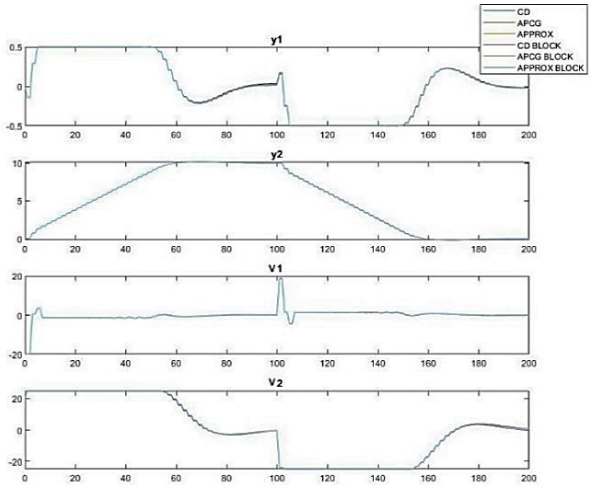

Fig. 1. Graphic comparison for the linear AFTI – 16 system.

The following tables present the comparison of the three algorithms, both on the scalar case and the block case, involving number of iterations for the inner and outer problems, as well as average and maximum CPU time for the outer problem and CPU time for one MPC step.

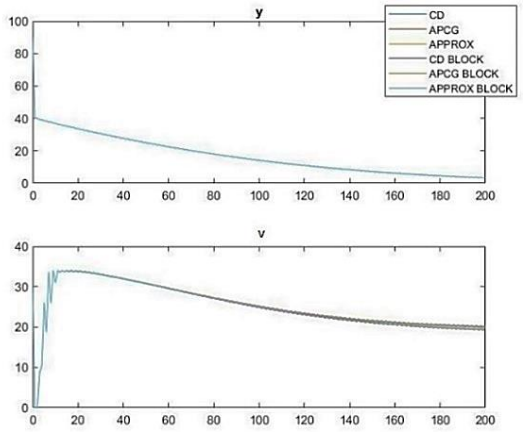

Fig. 2. Graphic comparison for the nonlinear WWTP system.

Table 1. Number of iterations and CPU time for the scalar case.

Syst. Algs.		A-16	H	M-S	I P	F T	WWTP	CSTR
CD	Inner it.	40115	32495	6365	2095	15620	20780	1085
	Outer it.	80	25	**40**	75	**15**	**5000**	**5000**
	MPC step	5.8	1.6	1.5	0.9	0.5	115.4	53.7
	Avg time	0.07	0.06	0.03	0.02	0.01	0.02	0.01
	Max time	0.1	0.4	0.09	0.1	0.02	0.07	0.04
APCG	Inner it.	5070	4730	5245	2252	4795	4060	2060
	Outer it.	80	**20**	**40**	**40**	**15**	**5000**	**5000**
	MPC step	**1.04**	**0.2**	**0.6**	**0.3**	**0.1**	**38.6**	**12.1**
	Avg time	**0.01**	**0.009**	**0.01**	**0.009**	**0.001**	**0.007**	**0.002**
	Max time	**0.01**	**0.1**	**0.03**	**0.04**	**0.002**	**0.03**	**0.03**
R-APPROX	Inner it.	**2000**	**2000**	**2250**	**2000**	**2000**	**2000**	**2000**
	Outer it.	**75**	**20**	**40**	**40**	**15**	**5000**	**5000**
	MPC step	2.04	0.3	0.9	0.6	0.2	87.7	34.9
	Avg time	0.02	0.01	0.02	0.01	0.02	0.01	0.007
	Max time	0.04	0.3	0.03	0.03	0.03	0.04	0.03

We observe that the number of iterations for the inner problem solved by the APCG and R – APPROX methods are significantly lower than the one obtained by applying the classical coordinate descent method, both in the scalar case and the block case. At the same time, all algorithms have significantly less inner iterations for the block case in comparison to the scalar case. Moreover, the CPU time is in all of the cases we take into consideration lower when the optimization problem is solved with APCG and R – APPROX than the CPU time obtained by solving it with the classic coordinate descent method. Observe that for very fast systems, like the Inverted pendulum, the block versions of the algorithms have a better average time than the scalar case versions.

Table 2. Number of iterations and CPU time for the block case.

Syst. Algs.		A-16	H	M-S	I P	F T	WWTP	CSTR
CD BLOCK	Inner it.	6113	3705	775	990	320	5020	325
	Outer it.	**75**	**20**	**40**	**40**	**15**	**5000**	**5000**
	MPC step	13.01	1.3	3.1	2.3	0.5	528.9	238.1
	Avg time	0.2	0.05	0.07	0.05	0.03	0.1	0.04
	Max time	0.3	0.1	0.4	0.1	0.1	0.2	0.1
APCG BLOCK	Inner it.	1175	946	**235**	**170**	**95**	1010	**135**
	Outer it.	**75**	**20**	**40**	**40**	**15**	**5000**	**5000**
	MPC step	**3.01**	**0.4**	**1.3**	**0.3**	**0.2**	**132.3**	**28.5**
	Avg time	**0.03**	**0.01**	**0.02**	**0.005**	**0.008**	**0.02**	**0.005**
	Max time	**0.04**	**0.04**	0.2	**0.02**	**0.02**	**0.06**	**0.04**
R-APPROX BLOCK	Inner it.	**550**	**550**	500	500	500	**550**	550
	Outer it.	**75**	**20**	**40**	**40**	**15**	**5000**	**5000**
	MPC step	9.9	1.01	2.8	1.9	**0.2**	439.4	52.8
	Avg time	0.1	0.04	0.06	0.04	0.01	0.08	0.01
	Max time	0.2	0.09	**0.09**	0.07	0.04	0.1	0.1

When it comes to comparing APCG and R – APPROX, the number of iterations and CPU time are lower for APCG compared to R – APPROX. However, one notable advantage of the R – APPROX method is that, unlike the APCG method, it doesn't depend on the value of the strong convexity parameter μ.

6. CONCLUSIONS

This paper has analyzed the performance of augmented Lagrangian framework to find the solution of MPC problems and several coordinate descent algorithms, namely the classic coordinate descent method, APCG and R – APPROX, are adapted to solve the inner problem. Both the APCG and R – APPROX converge faster than the classic coordinate descent method in all test cases we have considered. However, the APCG requires the value of the strong convexity parameter. Thus, despite APCG being faster than R – APPROX, one notable advantage of the latter is that, unlike the former, the restarted APPROX doesn't depend on the value of the strong convexity parameter μ. Thus, the method works with an unknown strong convexity parameter.

Acknowledgements: The research leading to these results has received funding from the NO Grants 2014-2021, under project ELO-Hyp, contract no. 24/2020 and was supported by the project "DINAMIC", Contract no. - 12PFE/2021.

REFERENCES

Annamalai, R. (2015). Discrete time Linear Quadratic (LQ) optimal control Vs MPC: Integral action and handling constraints (Master thesis). *Høgskolen i Telemark*

Bemporad, A. (2018). A Numerically Stable Solver for Positive Semidefinite Quadratic Programs Based on Nonnegative Least Squares. *IEEE Trans. Autom. Control*

63, 525–531, doi: https:// doi.org / 10.1109 /TAC.2017.2735938

Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N., (2002). The explicit linear quadratic regulator for constrained systems. *Automatica* 38, 3–20. doi: https://doi.org/10.1016/S0005-1098(01)00174-1

Bequette, B.W. (1998). Process dynamics: modeling, analysis, and simulation, Prentice Hall international series in the physical and chemical engineering sciences. *Prentice Hall PTR,* Upper Saddle River, N.J.

Bertsekas, D.P. (1997). Nonlinear Programming. *J. Oper. Res*. Soc. 48, 334–334. doi: https:// doi.org / 10.1057 /palgrave.jors. 2600425

Bertsekas, D. P. (1982). Chapter 2 - The Method of Multipliers for Equality Constrained Problems, in: Bertsekas, D.P. (Ed.), Constrained Optimization and Lagrange Multiplier Methods. *Academic Press*, 95–157. doi: https://doi.org/10.1016/B978-0-12-093480-5.50006-4

Bertsekas, D. P. (1982). Constrained Optimization and Lagrange Multiplier Methods. *Academic Press.* doi: https://doi.org/10.1016/B978-0-12-093480-5.50006-4

Caraman, S., Sbarciog, M., Barbu, M. (2006). Predictive control of a wastewater treatment process. *IFAC Proc. Vol., 1st IFAC Workshop on Applications of Large-Scale Industrial Systems* 39, 155–160. doi: https://doi.org/10.3182/20060830-2-SF-4903.00028

Chang, K.-W., Hsieh, C.-J., Lin, C.-J. (2008). Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines. *Journal of Machine Learning Research* 9, 1369–1398. doi: 10.1145/1390681.1442778

Fercoq, O., Qu, Z. (2020). Restarting the accelerated coordinate descent method with a rough strong convexity estimate. *Computational Optimization and Applications*. 75, 63–91. doi: https:// doi.org /10.1007 /s10589-019-00137-2

Frank, M., Wolfe, P. (1956). An algorithm for quadratic programming. *Nav. Res. Logist. Q.* 3, 95–110. doi: https://doi.org/10.1002/nav.3800030109

Gondzio, J., Grothey, A. (2007). Parallel interior-point solver for structured quadratic programs: Application to financial planning problems. *Ann. Oper. Res.* 152, 319–339. doi: https://doi.org/10.1007/s10479-006-0139-z

He, B., Yuan, X. (2010). On the acceleration of augmented Lagrangian method for linearly constrained optimization, *Computer Science*. https://optimization-online.org/2010/10/2760/

Hestenes, M.R. (1969). Multiplier and gradient methods. *J. Optim. Theory Appl.* 4, 303–320. doi: https://doi.org/10.1007/BF00927673

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM, in: Proceedings of the 25th International Conference on Machine Learning - ICML '08. Presented at the the 25th international conference, *ACM Press*, Helsinki, Finland, 408–415, doi: https:// doi.org/ 10.1145 /1390156.1390208

Kang, M., Kang, M., Jung, M. (2015). Inexact accelerated augmented Lagrangian methods. *Computational Optimization and Applications* 62, 373–404. doi: https://doi.org/10.1007/s10589-015-9742-8

Kapasouris, P., Athans, M., Stein, G. (1988). Design of feedback control systems for stable plants with saturating actuators, in: Proceedings of the 27th IEEE Conference on Decision and Control. Presented at the Proceedings of the 27th IEEE Conference on Decision and Control 1, 469–479. doi: https://doi.org/10.1109/CDC.1988.194356

Kvasnica, M. (2016). Implicit vs explicit MPC — Similarities, differences, and a path towards a unified method, in: 2016 European Control Conference (ECC). Presented at the 2016 European Control Conference (ECC), 603–603, doi: https:// doi.org /10.1109 /ECC. 2016. 7810353

Lin, Q., Lu, Z., Xiao, L. (2014). An Accelerated Proximal Coordinate Gradient Method and its Application to Regularized Empirical Risk Minimization. arXiv 1407.1296v1. doi: https:// doi.org /10.48550 /arXiv. 1407.1296

Longo, S., Kerrigan, E.C., Constantinides, G.A. (2013). A predictive control solver for low-precision data representation, in: 2013 European Control Conference (ECC). Presented at the 2013 European Control Conference (ECC), 3590–3595. doi: https://doi.org/10.23919/ECC.2013.6669255

Maeder, U., Cagienard, R., Morari, M. (2007). Explicit Model Predictive Control, in: Tarbouriech, S., Garcia, G., Glattfelder, A.H. (Eds.), Advanced Strategies in Control Systems with Input and Output Constraints, Lecture Notes in Control and Information Sciences. *Springer,* Berlin, Heidelberg, 237–271. doi: https://doi.org/10.1007/978-3-540-37010-9_8

Necoara, I., Clipici, D. (2015). Parallel coordinate descent methods for composite minimization: convergence analysis and error bounds. arXiv:1312.5302v4, doi: https://doi.org/10.48550/arXiv.1312.5302

Nedelcu, V., Necoara, I., Tran-Dinh, Q. (2013). Computational Complexity of Inexact Gradient Augmented Lagrangian Methods: Application to Constrained MPC. *SIAM J. Control Optim.* 52(5). doi: https://doi.org/10.1137/120897547

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Math. Program.* 103, 127–152. doi: https://doi.org/10.1007/s10107-004-0552-5

Nesterov, Y. (2012). Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems. *SIAM J. Optim.* 22, 341–362. doi: https://doi.org/10.1137/100802001

Richtárik, P., Takáč, M. (2016). Distributed Coordinate Descent Method for Learning with Big Data. *Journal of Machine Learning Research* 17, 2657–268.

Tøndel, P., Johansen, T.A. (2002). Complexity reduction in linear Model Predictive Control, IFAC Proc. Vol., 15th IFAC World Congress 35, 189–194. doi: https://doi.org/10.3182/20020721-6-ES-1901.00600

Wu, L., Bemporad, A. (2022). A Simple and Fast Coordinate-Descent Augmented-Lagrangian Solver for Model Predictive Control. arXiv:2109.10205v3 doi: https://doi.org/10.48550/arXiv.2109.10205