Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 56-2 (2023) 9423–9428

# A construction-free coordinate-descent augmented-Lagrangian method for embedded linear MPC based on ARX models

**Liang Wu** *, **Alberto Bemporad** *

* *IMT School for Advanced Studies Lucca, Italy, (e-mail: {liang.wu,alberto.bemporad}@imtlucca.it)*

**Abstract:** This paper proposes a construction-free algorithm for solving linear MPC problems based on autoregressive with exogenous terms (ARX) input-output models. The solution algorithm relies on a coordinate-descent augmented Lagrangian (CDAL) method previously proposed by the authors, which we adapt here to exploit the special structure of ARX-based MPC. The CDAL-ARX algorithm enjoys the construction-free feature, in that it avoids explicitly constructing the quadratic programming (QP) problem associated with MPC, which would eliminate construction cost when the ARX model changes/adapts online. For example, the ARX model parameters are dependent on linear parameter-varying (LPV) scheduling signals, or recursively adapted from streaming input-output data with cheap computation cost, which make the ARX model widely used in adaptive control. Moreover, the implementation of the resulting CDAL-ARX algorithm is matrix-free and library-free, and hence amenable for deployment in industrial embedded platforms. We show the efficiency of CDAL-ARX in two numerical examples, also in comparison with MPC implementations based on other general-purpose quadratic programming solvers.



*Keywords:* ARX, State-Space, Model Predictive Control, Construction-free

## 1. INTRODUCTION

Model Predictive Control (MPC) is an advanced technique to control multi-input multi-output systems subject to constraints, and its core idea is to predict the evolution of the controlled system by means of a dynamical model, solve an optimization problem over a finite time horizon, only implement the control input at the current time, and then repeat the optimization again at the next sample, see Qin and Badgwell (2003).

In earlier MPC developments, some methods shared the same receding horizon control idea, under different names. The Model Predictive Heuristic Control, the Model Algorithmic Control used a finite impulse response model, the Dynamic Matrix Control (DMC) employed a truncated step-response model, and the Generalized Predictive Control (GPC) involved a transfer function model. As the MPC field has grown, state-space (SS) models replaced input-output (I/O) models, and most MPC theory is based on SS formulations, see Mayne (2014).

However, in industrial control applications, MPC based on input-output models, such as the autoregressive model with exogenous terms (ARX) model, may still be preferable, see Qin and Badgwell (2003), for two main reasons: (1) there is no need of a state-observer; (2) I/O models are easier to identify and to adapt online (such as using recursive least-squares or Kalman Filter algorithms), which makes them widely used in adaptive control, see Åström and Wittenmark (2013). In particular, the latter is particularly appealing in practical cases in which the dynamics of the systems changes during operations, such as in the case of changes of mass and inertia in rockets due to fuel consumption, wear of heating equipment in chemical processes, and many others. In fact, an observable SS model can be equivalently transformed into an ARX model, and Wu (2022a) shows the equivalence of SS-based MPC and ARX-based MPC problems. It proposes an alternative for the acquisition of ARX model based on the first-principle-based modeling paradigm, rather than the data-driven identification paradigm. This allows us to acquire ARX models using many existing first-principles based models in different engineering fields. The resulted interpretative ARX model can be adopted in adaptive MPC framework by adding an online updating scheme for the ARX model, see Wu (2022b).

A common practice in MPC is to first formulate a quadratic programming (QP) problem in terms of a control-oriented prediction model and MPC parameters, and then pass it to the optimization solver. Such a problem construction step can be performed offline when the prediction model is fixed, otherwise, it requires to be repeated online when the prediction model or MPC parameters are varying. In such varying cases, the online computation time includes both constructing and solving the QP problem associated with MPC. Indeed, often constructing and solving the MPC problem have comparable costs, such as when warm-starting strategies are employed and set-points change slowly. In Wu (2022b), the online construction


10.1016/j.ifacol.2023.10.235

of the MPC problem becomes necessary in the adaptive ARX-based MPC framework, and the case of linear parameter varying ARX (LPV-ARX) models, in which model parameters depend on a measured time-varying signal, the so-called scheduling variable. Thus, a construction-free ARX-MPC algorithm, in that MPC-to-QP construction is explicitly eliminated, would significantly save the computational loads in those cases.

### *1.1 Related works and Contribution*

Some ARX-based MPC algorithms in the literature first convert the ARX model into SS form, treat the problem as a standard SS-based MPC problem, see Huusom et al. (2010), and then construct and solve a condensed or sparse quadratic programming (QP) problem. In fact, the ARX-to-SS transformation is not necessary for condensed and sparse MPC-to-QP constructions, which only depend on whether to eliminate or keep the ARX output variables. Choosing the condensed or sparse construction only depends on the total online computation cost (constructing and solving), in time-varying ARX-based MPC problems. The OSQP solver, based on the alternating direction method of multipliers (ADMM), can directly consume the ARX model as equality constraints, which is the sparse QP formulation by keeping the output variables of the ARX model. However, it still employs an explicit MPC-to-QP construction to formulate the equality constraint matrix, and more importantly, the OSQP solver needs to repeatedly factorize and cache the Hessian matrix of the quadratic objective at each sampling time, in time-varying ARX-based MPC problems. In Saraf and Bemporad (2017), the dynamic equality constraint from the ARX model was relaxed by using a large penalty parameter, and it resulted in an ill-conditioning bounded variable least-squares (BVLS) problem, although the active-set based method was used to mitigate the numerical difficulties to some extent. Besides computation efficiency, easy-to-deployment of an ARX-MPC algorithm should also be considered, in which code-simplicity and library-dependency are important. In this respect, compared to the active-set or interior-point based methods, the first-order method, such as the primal or dual fast gradient method, the ADMM method, is simpler but also becomes complicated in time-varying cases, in that some offline operations have to be performed online.

This paper proposes a simple and efficient algorithm for solving ARX-based MPC problems. Based on the coordinate-descent augmented Lagrangian method, the resulting CDAL-ARX algorithm enjoys three mian features: (*i*) it is *construction-free*, in that it avoids the online MPC-to-QP construction in time-varying ARX cases to save computation cost; (*ii*) is *matrix-free*, in that it avoids multiplications and factorizations of matrices, which are required by other first-order methods in time-varying ARX cases; and (*iii*) is *library-free*, as our 150-lines of C-code implementation is without any library dependency, which matters in embedded deployment.

## 2. ARX-BASED MPC PROBLEM FORMULATION

Consider the multi-input multi-output (MIMO) ARX model described by

$$y_t = \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i} \tag{1}$$

where $y_t \in \mathbb{R}^{n_y}$ and $u_t \in \mathbb{R}^{n_u}$ are the output and input of the system, respectively, $A(i) \in \mathbb{R}^{n_y \times n_y}$, $i = 1, \dots, n_a$, and $B(i) \in \mathbb{R}^{n_y \times n_u}$, $i = 1, \dots, n_b$, and $n_a$, $n_b$ define the model order of the ARX model.

This paper considers the following MPC tracking formulation based the ARX model (1)

$$\begin{aligned} \min_{Y,U,\Delta U} \ & \frac{1}{2} \sum_{t=1}^{T} \|(y_t - r_t)\|^2_{W^y} + \|\Delta u_{t-1}\|^2_{W^{\Delta u}} \\ \text{s.t} & y_t = \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i=1}^{n_b} B(i) u_{t-i}, \ t = 1, \dots, T \\ & \Delta u_t = u_t - u_{t-1}, \ t = 0, \dots, T-1 \\ & y_{\min} \le y_t \le y_{\max}, \ t = 1, \dots, T \\ & u_{\min} \le u_t \le u_{\max}, \ t = 0, \dots, T-1 \\ & \Delta u_{\min} \le \Delta u_t \le \Delta u_{\max}, t = 0, \dots, T-1 \end{aligned} \tag{2}$$

where $T$ is the prediction horizon, $W^y \succeq 0$ and $W^{\Delta u} \succeq 0$ are positive semi-definiteness diagonal matrices on the outputs and the input increments, respectively, $r_t$, $t = 1, \dots, T$ are the future desired set-point vectors, $\Delta u_{t-1}$ are the input increments, $[y_{\min}, y_{\max}]$, $[u_{\min}, u_{\max}]$, and $[\Delta u_{\min}, \Delta u_{\max}]$ define box constraints on outputs, inputs, and input increments, respectively, and $Y = (y_1, \dots, y_T)$, $U = (u_0, \dots, u_{T-1})$, and $\Delta U = (\Delta u_0, \dots, \Delta u_{T-1})$ are the optimization variables.

## 3. COORDINATE DESCENT AUGMENTED LAGRANGIAN METHOD

Wu and Bemporad (2023) proposed a coordinate-descent augmented-Lagrangian (CDAL) method for SS-based MPC problems. We want to adapt here the method to solve problem (2) without computing a state-space realization of the ARX model (1), while retaining the construction-free, matrix-free, and library-free properties of CDAL.

### *3.1 Augmented Lagrangian method*

The following assumptions are needed to ensure the convergence of the Augmented Lagrangian method.

*Assumption 1.* Problem (2) has a feasible solution.

Note that Assumption (1) is satisfied in all practical situations in which the reference $r_t$ is far enough from the output bounds and the prediction horizon $T$ is long enough.

*Assumption 2.* The equality constraint matrix arising from stacking all the equality constraints in (2) is full rank at the optimal solution of the problem.

Let $\mathcal{Y}$, $\mathcal{U}$, and $\Delta\mathcal{U}$ denote the hyper-boxes on $Y$, $U$, and $\Delta U$, respectively, defined by the box constraints in (2), respectively. The bound-constrained Augmented Lagrangian function $\mathcal{L}_\rho : \mathcal{Y} \times \mathcal{U} \times \Delta\mathcal{U} \times \mathbb{R}^{Tn_y} \times \mathbb{R}^{Tn_u} \to \mathbb{R}$ is given by

$$\begin{aligned}\mathcal{L}_\rho(Y,U,\Delta U,\Lambda,\Gamma) &= \frac{1}{2}\sum_{t=1}^{T}\|(y_t-r_t)\|^2_{Wy}+\|\Delta u_{t-1}\|^2_{W\Delta u}\\ &+\sum_{t=1}^{T}\lambda_t'\left(\sum_{i=1}^{n_a}A(i)y_{t-i}+\sum_{i=1}^{n_b}B(i)u_{t-i}-y_t\right)\\ &+\sum_{t=1}^{T}\gamma_t'\left(u_{t-2}+\Delta u_{t-1}-u_{t-1}\right)\\ &+\frac{\rho}{2}\sum_{t=1}^{T}\left\|\sum_{i=1}^{n_a}A(i)y_{t-i}+\sum_{i=1}^{n_b}B(i)u_{t-i}-y_t\right\|_2^2\\ &+\frac{\rho}{2}\sum_{t=1}^{T}\|u_{t-2}+\Delta u_{t-1}-u_{t-1}\|_2^2\end{aligned} \tag{3}$$

where $\Lambda=\{\lambda_t\}$ and $\Gamma=\{\gamma_t\},\forall t=1,\dots,T$ are the dual vectors associated with the equality constraints induced by the ARX model and the input increments, respectively, and $\rho$ is the penalty parameter. According to Bertsekas (2014), the scaled AL method (ALM) iterates the following updates

$$(Y^k,U^k,\Delta U^k)=\operatorname{argmin}\frac{1}{\rho}\mathcal{L}_\rho(Y,U,\Delta U,\Lambda^{k-1},\Gamma^{k-1}) \tag{4a}$$

$$\lambda_t^k=\lambda_t^{k-1}+\sum_{i=1}^{n_a}A(i)y_{t-i}^k+\sum_{j=1}^{n_b}B(i)u_{t-i}^k-y_t^k \tag{4b}$$
$$,\forall t=1,\dots,T$$

$$\gamma_t^k=\gamma_t^{k-1}+u_{t-1}^k+\Delta u_t^k-u_t^k,\forall t=1,\dots,T \tag{4c}$$

The minimization step (4a) updates the primal vector, Steps (4b) and (4c) update the dual vectors. We refer the reader to Bertsekas (2014) for a well-known convergence proof of ALM under Assumptions 1, 2. To improve the speed of convergence of ALM, Kang et al. (2015) proposed an accelerated version of ALM whose convergence rate is $O(1/k^2)$ for linearly constrained convex programs by using Nesterov's acceleration technique, see Nesterov (1983). The accelerated ALM algorithm for MPC problems has been summarized in Wu and Bemporad (2023).

### *3.2 Coordinate-descent method*

Sub-problem (4a) is a strongly convex box-constrained QP problem, which can be solved by many methods. Among others, as showed in Wu and Bemporad (2023), problem (4a) can be solved by a simple coordinate-descent method, which minimizes the objective function along only one coordinate direction at each iteration while keeping the other coordinates fixed, see Luo and Tseng (1992). A convergence proof of at-least linear convergence when solving convex differentiable minimization problems was provided in Luo and Tseng (1992). Under Assumption 1 (non-emptyness of the feasible set) and since the objective function $\mathcal{L}_\rho(\cdot)$ is continuously differentiable and convex with respect to each coordinate, the CD method proceeds repeatedly for $k=1,2,\dots,$ as follows:

$$\text{choose } j_k\in\{1,2,\dots,n_z\} \tag{5a}$$

$$z_{j_k}^k=\operatorname*{argmin}_{z_{j_k}\in\mathcal{Z}}\frac{1}{\rho}\mathcal{L}_\rho(z_{j_k},z_{\neq j_k}^{k-1},\Lambda^{k-1},\Gamma^{k-1}) \tag{5b}$$

**Procedure 1** Full pass of cyclic coordinate descent on all block variables

**Input**: $\Lambda=\{\lambda_1,\dots,\lambda_T\}$, $\Gamma=\{\gamma_1,\dots,\gamma_T\}$, $Y=\{y_1,\cdots,y_T\}$, $U=\{u_0,\cdots,u_{T-1}\}$, $\Delta U=\{\Delta u_0,\cdots,\Delta u_{T-1}\}$; MPC settings $A(1),\dots,A(n_a)$, $B(1),\dots,B(n_b)$, $W^y$, $W^{\Delta u}$, $y_{\min}$, $y_{\max}$, $u_{\min}$, $u_{\max}$, $\Delta u_{\min}$, $\Delta u_{\max}$; parameter $\sigma,\rho>0$.

1. $\sigma\leftarrow 0$;
2. **for** $t=1,\dots,T-1$ **do**
   - 2.1. $j=\min(n_a,T-t)$;
   - 2.2. $\{y_t,\sigma\}\leftarrow \text{CCD}\{\frac{1}{\rho}W^y+I+\sum_{i=1}^{j}A(i)'A(i),e_t,\sigma\}_{y_{\min}}^{y_{\max}}$;
   - 2.3. $j=\min(n_b,T-t+1)$;
   - 2.4. $\{u_{t-1},\sigma\}\leftarrow \text{CCD}\{2I+\sum_{i=1}^{j}B(i)'B(i),f_t,\sigma\}_{u_{\min}}^{u_{\max}}$;
   - 2.5. $\{\Delta u_t,\sigma\}\leftarrow \text{CCD}\{\frac{1}{\rho}W^{\Delta u}+I,g_t,\sigma\}_{\Delta u_{\min}}^{\Delta u_{\max}}$;
3. $\{y_T,\sigma\}\leftarrow \text{CCD}\{\frac{1}{\rho}W^y+I,e_T,\sigma\}_{y_{\min}}^{y_{\max}}$;
4. $\{u_{T-1},\sigma\}\leftarrow \text{CCD}\{I+B(1)'B(1),f_T,\sigma\}_{u_{\min}}^{u_{\max}}$;
5. $\{\Delta u_{T-1},\sigma\}\leftarrow \text{CCD}\{\frac{1}{\rho}W^{\Delta u}+I,g_T,\sigma\}_{\Delta u_{\min}}^{\Delta u_{\max}}$;
6. **end**.

**Output**: $Y$, $U$, $\Delta U$, $\Lambda$, $\Gamma$, $\sigma$.

where $z=\begin{bmatrix}y_1' & u_0' & \Delta u_0' & \dots & y_T' & u_{T-1}' & \Delta u_{T-1}'\end{bmatrix}'$ is the optimization vector, $z\in\mathcal{Z}\triangleq\mathcal{Y}\times\mathcal{U}\times\Delta\mathcal{U}$, $\mathcal{Z}\subseteq\mathbb{R}^{n_z}$, $n_z\triangleq T(n_y+n_u+n_u)$. We denote by $\mathcal{L}_\rho(z_{j_k},z_{\neq j_k}^{k-1},\Lambda^{k-1},\Gamma^{k-1})$ the value $\mathcal{L}_\rho(z,\Lambda^{k-1},\Gamma^{k-1})$ when $z_{\neq j_k}=z_{\neq j_k}^k$ is fixed. Here $z_{\neq j_k}$ denotes the subvector obtained from $z$ by eliminating its $j_k$th component $z_{j_k}$. The convergence of the iterations (5) depends on the coordinate picking rule, namely how $j_k$ is chosen. Existing research works have analyzed the influence of different coordinate selection rules such as the cyclic rule and the random selection rule, on the convergence rate of the coordinate descent method. We choose the simplest variant using cyclic coordinate search to favor implementation simplicity. In fact, the cyclic implementation preserves the order of optimization variables with respect to the prediction horizon $t$, type (output $y$, input $u$, or input increment $\Delta u$), and component, so that reconstructing the coordinate that is currently optimized in is immediate. The implementation of one pass through all $n_z$ coordinates using cyclic CD is reported in Procedure 1. In Procedure 1, the operator $\{s,\sigma\}=\text{CCD}\{M,d,\sigma\}_{\underline{s}}^{\bar{s}}$ represents one pass iteration of the reverse cyclic CD method through all its $n_s$ coordinates $s_1,\dots,s_{n_s}$ for the box-constrained QP $\min_{s\in[\underline{s},\bar{s}]}\frac{1}{2}s'Ms+s'd$, that is to execute the following $n_s$ iterations

$$\begin{aligned}&\text{for } i=1,\dots,n_s\\ &\quad\hat{s}_i\leftarrow\max(\underline{s}_i,\min(\bar{s}_i,s_i-\frac{1}{M_{i,i}}(M_{i,\cdot}s+d_i))\\ &\quad\sigma\leftarrow\sigma+(\hat{s}_i-s_i)^2\\ &\quad s_i\leftarrow\hat{s}_i\\ &\text{end}\end{aligned} \tag{6}$$

The quantities $e_t,f_t,g_t$ used in Procedure 1 are defined for $t=1,2,\cdots,T$ as follows:

$$
\begin{aligned}
e_t &= -W^y r - (\lambda_t + \sum_{i=1}^{n_a} A(i) y_{t-i} + \sum_{i}^{n_b} B(i) u_{t-i}) \\
&\quad + \sum_{n_i=1}^{\min(n_a, T-t)} A(n_i)' (\lambda_{t+n_i} + \sum_{i \neq n_i}^{n_a} A(i) y_{t+n_i-i} \\
&\quad + \sum_{i=1}^{n_b} B(i) u_{t+n_i-i} - y_{t+n_i}) \\
f_t &= -(\gamma_t + u_{t-2} + \Delta u_{t-1}) + (\gamma_{t+1} + \Delta u_t + u_t) \\
&\quad + \sum_{n_i=1}^{\min(n_b, T-t+1)} B(n_i)' (\lambda_{t+n_i} + \sum_{i=1}^{n_a} A(i) y_{t+n_i-i} \\
&\quad + \sum_{i \neq n_i}^{n_b} B(i) u_{t+n_i-i} - y_{t+n_i}) \\
g_t &= \gamma_t + u_{t-2} - u_{t-1}
\end{aligned}
$$

$$
\begin{aligned}
e_T &= -W^y r - (\lambda_T + \sum_{i=1}^{n_a} A(i) y_{T-i} + \sum_{i=1}^{n_b} B(i) u_{T-i}) \\
f_T &= -(\gamma_T + u_{T-2} + \Delta u_{T-1}) + B(1)' (\lambda_T \\
&\quad + \sum_{i=1}^{n_a} A(i) y_{T-i} + \sum_{i \neq 1}^{n_b} B(i) u_{T-i} - y_T) \\
g_T &= \gamma_T + u_{T-2} - u_{T-1}
\end{aligned}
$$

which shows that they involve several matrix-vectors multiplications. It would greatly affect the computation efficiency since their computational cost is proportional to the product of the inner iterations and the outer iterations. To eliminate their explicit calculation, we propose here below an efficient coupling scheme between CD and AL that reduces the cost per iteration, without changing the rate of convergence of the algorithm.

### *3.3 Efficient coupling scheme between CD and AL*

Our proposed efficient coupling scheme exploits the fact that CD only updates one coordinate each time, and the execution (6) of the operator CCD(·) involves the next update of dual Lagrangian vectors. Here we take Step 2.2 of Procedure 1 as an example, which has been modified from equation (6) to Procedure 2. Note that the dual Lagrangian vectors used in Procedure 2 have been updated before Procedure 2. The symbols $\{D_0^y, D_1^y, \ldots, D_{T-1}^y\}$ denote the diagonal elements of their Hessian matrices used in Step 2.2 and 3

$$
\begin{aligned}
&\text{for } t = 1, \ldots, T-1 \\
&\quad j = \min(n_a, T-t); \\
&\quad D_t^y \leftarrow \operatorname{diag}\left(\frac{1}{\rho} W^y + I + \sum_{i=1}^{j} A(i)' A(i)\right) \qquad (7) \\
&\text{end} \\
&D_{T-1}^y \leftarrow \frac{1}{\rho} W^y + I
\end{aligned}
$$

To avoid repeating division operations, the values $\{\frac{1}{D_0^y}, \frac{1}{D_1^y}, \ldots, \frac{1}{D_{T-1}^y}\}$ are cached before the iterations start. The other steps involving the operator CCD(·) in Procedure 1 follow the same idea.

**Procedure 2** One pass of cyclic coordinate descent for Step 2.2 of Procedure 1 after using efficient coupling scheme

**Input**: $j = \min(n_a, T-t)$; $y_{t+1}$, $\lambda_t$, $\lambda_{t+1}, \ldots, \lambda_{t+j}$; parameter $\rho > 0$; update amount $\sigma \geq 0$.

1. **for** $i = 1, \ldots, n_y$ **do**
   - 1.1. $s \leftarrow -\lambda_{t,i} + \sum_{n_i=1}^{j} A(n_i)'_{:,i} \lambda_{t+n_i}$;
   - 1.2. $\theta \leftarrow \left[ y_{t,i} - \frac{\frac{1}{\rho} W_i^y (y_{t,i} - r_i) + s}{D_{t,i}^y} \right]_{y_{\min,i}}^{y_{\max,i}}$;
   - 1.3. $\Delta \leftarrow \theta - y_{t,i}$;
   - 1.4. $\sigma \leftarrow \sigma + \Delta^2$;
   - 1.5. $y_{t,i} \leftarrow \theta$;
   - 1.6. $\lambda_{t,i} \leftarrow \lambda_{t,i} + \Delta$;
   - 1.7. **for** $n_i = 1, \ldots, j$ **do**
     - 1.7.1. $\lambda_{t+n_i} \leftarrow \lambda_{t+n_i} + \Delta \cdot A(n_i)_{:,i}$
2. **end**.

**Output**: $y_t$, $\lambda_t$, $\lambda_{t+1}, \ldots, \lambda_{t+j}$, $\sigma$.

### *3.4 Algorithm*

Summarizing all the ingredients described in the previous sections, we obtain the construction-free ARX-based MPC Algorithm 3, which we call CDAL-ARX. Here, construction-free means that CDAL-ARX directly uses the ARX model coefficients without the need of constructing a QP problem explicitly. Note that the main update of the Lagrangian variables in Algorithm 3 is placed early in Step 2.1, which is different from the original version of Algorithm 1 in Wu and Bemporad (2023) because the CD method allows the use of our proposed efficient coupling scheme. The quantities $N_{\text{out}}$ and $N_{\text{in}}$ denote the maximum number of AL (outer-loop) and CD (inner-loop) iterations, respectively. The tolerances $\epsilon_{\text{out}}$ and $\epsilon_{\text{in}}$ define the stopping criteria of the outer and inner iterations, respectively.

## 4. NUMERICAL EXAMPLES

In this section, we test our proposed ARX-based MPC algorithm against other MPC solvers, which rely on condensed or sparse MPC-to-QP construction, respectively. The best choice between condensed and sparse QP forms mainly depends on the number of outputs $n_y$, control inputs $n_u$, and the length of the prediction horizon $T$, see Kouzoupis et al. (2015). For numerical comparisons with our ARX-based MPC algorithm, this paper considers both condensed and sparse MPC-to-QP constructions, which are then solved by the qpOASES (Ferreau et al. (2014)) and OSQP (Stellato et al. (2020)), respectively. The reported comparison simulation results were obtained on a MacBook Pro with a 2.7 GHz 4-core Intel Core i7 and 16GB RAM. Algorithm 3, qpOASES v3.2 and OSQP v0.6.2 are all executed in MATLAB R2020a via their C-mex implementations.

### *4.1 Problem descriptions*

(1) Time-varying ARX model example: one notable feature of ARX models is their ease to be updated at runtime, which makes them particularly appealing when the system dynamics cannot be well captured by a single linear time-invariant model. Our CDAL-ARX algorithm can take advantage of its construction-free feature to avoid the computation

**Algorithm 3** Accelerated cyclic CDAL algorithm for ARX-based MPC

**Input**: primal/dual warm-start $Y = \{y_1, y_2, \cdots, y_T\}$, $U = \{u_0, u_1, \cdots, u_{T-1}\}$, $\Delta U = \{\Delta u_0, \Delta u_1, \cdots, \Delta u_{T-1}\}$, $\Lambda^{-1} = \Lambda^0 = \{\lambda_1, \lambda_2, \cdots, \lambda_T\}$, $\Gamma^{-1} = \Gamma^0 = \{\gamma_1, \gamma_2, \cdots, \gamma_T\}$; History input and output data $\{y_0, y_{-1}, \ldots, y_{1-n_a}\}$, $\{u_{-1}, u_{-2}, \ldots, u_{1-n_b}\}$; MPC settings $\{A(1), A(2), \ldots, A(n_a)$, $B(1), B(2), \ldots, B(n_b)$, $W^y$, $W^{\Delta u}$, $y_{\min}$, $y_{\max}$, $u_{\min}$, $u_{\max}$, $\Delta u_{\min}$, $\Delta u_{\max}\}$; Algorithm settings $\{\rho, N_{\text{out}}, N_{\text{in}}\, \epsilon_{\text{out}}, \epsilon_{\text{in}}\}$

1. $\alpha_1 \leftarrow 1$; $\hat{\Lambda}^0 \leftarrow \Lambda^0$; $\hat{\Gamma}^0 \leftarrow \Gamma^0$;
2. **for** $k = 1, 2, \cdots, N_{\text{out}}$ **do**
   2.1. **for** $t = 1, 2, \cdots, T$ **do**
      2.1.1. $\lambda_t^k = \hat{\lambda}_t^{k-1} + (\sum_{i=1}^{n_a} A(i) y_{t-i}^k + \sum_{j=1}^{n_b} B(i) u_{t-i}^k - y_t^k)$
      2.1.2. $\gamma_t^k = \hat{\gamma}_t^{k-1} + (u_{t-2}^k + \Delta u_{t-1}^k - u_{t-1}^k)$;
   2.2. **for** $k_{in} = 1, 2, \cdots, N_{\text{in}}$ **do**
      2.2.1. $(Y, U, \Delta U, \sigma) \leftarrow$ Procedure 1 with use of Procedure 2;
      2.2.2. **if** $\sigma \le \epsilon_{\text{in}}$ **break** the loop;
   2.3. **if** $\|\Lambda^k - \hat{\Lambda}^{k-1}\|_2^2 + \|\Gamma^k - \hat{\Gamma}^{k-1}\|_2^2 \le \epsilon_{\text{out}}$ **stop**;
   2.4. $\alpha_{k+1} \leftarrow \frac{1+\sqrt{1+4\alpha_k^2}}{2}$;
   2.5. $\hat{\Lambda}^k \leftarrow \Lambda^k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\Lambda^k - \Lambda^{k-1})$;
3. **end**.

**Output**: $Y, U, \Delta U, \Lambda, \Gamma$

cost of the online construction step. We tested CDAL-ARX on randomly-generated two-input-two-output ARX models with order $n_a = 4$ and $n_b = 4$ and time-varying system matrices. For demonstration purposes, here below we report one instance of them, whose ARX coefficient matrices $A^t(1), \ldots, A^t(4)$, $B^t(1), \ldots, B^t(4)$ at time $t$ are given by

$$\begin{aligned} A(i)^t &= A(i) + 0.1M^t, i = 1, \ldots 4 \\ B(i)^t &= B(i) + 0.1M^t, i = 1, \ldots 4 \end{aligned} \tag{8}$$

where $A(1) = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$, $A(2) = \begin{bmatrix} 0.7 & 0.1 \\ 0.1 & 0.7 \end{bmatrix}$, $A(3) = \begin{bmatrix} 0.5 & 0.1 \\ 0.1 & 0.5 \end{bmatrix}$, $A(4) = \begin{bmatrix} 0.3 & 0.1 \\ 0.1 & 0.3 \end{bmatrix}$, $B(1) = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$, $B(2) = \begin{bmatrix} 0.8 & 0.4 \\ 0.4 & 0.8 \end{bmatrix}$, $B(3) = \begin{bmatrix} 0.6 & 0.3 \\ 0.3 & 0.6 \end{bmatrix}$, $B(4) = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 0.4 \end{bmatrix}$, $M^t = \begin{bmatrix} \sin(\frac{t}{10}) & \cos(\frac{t}{10}) \\ \cos(\frac{t}{10}) & \sin(\frac{t}{10}) \end{bmatrix}$.

(2) *DNN-based LPV-ARX model example*: We tested on CDAL-ARX on randomly-generated two-input-two-output quasi-LPV-ARX models of larger order $n_a = 6$ and $n_b = 6$, whose coefficient matrices are piecewise affine (PWA) maps of the scheduling vector $w_{t-1}$

$$\begin{bmatrix} y_t(1) \\ y_t(2) \end{bmatrix} = \begin{bmatrix} \mathcal{N}_1(w_{t-1})' \\ \mathcal{N}_2(w_{t-1})' \end{bmatrix} x_{t-1} \tag{9}$$

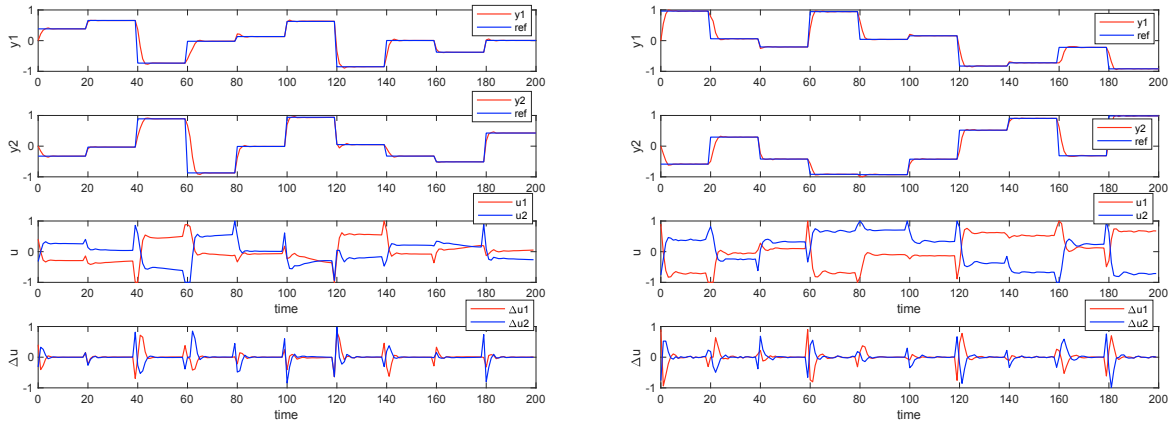
where $x_{t-1} = \left[y_{t-1}', \ldots, y_{t-6}', u_{t-1}', \ldots, u_{t-6}'\right]' \in \mathbb{R}^{24}$, $w_{t-1} = \left[y_{t-1}', \ldots, y_{t-6}', u_{t-2}', \ldots, u_{t-6}'\right]' \in \mathbb{R}^{22}$, and $\mathcal{N}_1, \mathcal{N}_2 \in \mathbb{R}^{22} \to \mathbb{R}^{24}$ are deep feedforward neural networks with three layers and ReLU activation function, namely $\mathcal{N}_1(w_{t-1}) = W_{1,3} \max(0, W_{1,2} \max(0, W_{1,1} w_{t-1} + b_{1,1}) + b_{1,2}) + b_{1,3}$, $\mathcal{N}_2(w_{t-1}) = W_{2,3} \max(0, W_{2,2} \max(0, W_{2,1} w_{t-1} + b_{2,1}) + b_{2,2}) + b_{2,3}$. Here we choose the number of neurons in each hidden layer as three times the number of inputs according to Serra et al. (2018), that is, $W_{1,1}$ and $W_{2,1} \in \mathbb{R}^{66 \times 22}$, $b_{1,1}$ and $b_{2,1} \in \mathbb{R}^{66}$, $W_{1,2}$ and $W_{2,2} \in \mathbb{R}^{66 \times 66}$, $b_{1,2}$ and $b_{2,2} \in \mathbb{R}^{66}$, $W_{1,3}$ and $W_{2,3} \in \mathbb{R}^{24 \times 66}$, $b_{1,3}$ and $b_{2,3} \in \mathbb{R}^{24}$. For demonstration purposes, we define $b_{1,3}, b_{2,3}$ by collecting the coefficients defining $A(1), \ldots, A(4), A(4), A(4), B(1), \ldots, B(4), B(4), B(4)$ as in (8); the remaining network parameters are randomly generated uniformly between 0 and 0.1. At each time $t$, the linear model consumed by our CDAL-ARX algorithm is given by evaluating the deep ReLU networks as in (9).

In both examples, we use the same MPC parameters $W_y = I$, $W_{\Delta u} = 0.1I$, $[y_{\min}, y_{\max}] = [-1, 1]$, $[u_{\min}, u_{\max}] = [-1, 1]$, $[\Delta u_{\min}, \Delta u_{\max}] = [-1, 1]$. Different prediction horizon lengths $T$ are used to investigate numerical performance, namely $T = 10$, 20, and 30. Their history input-output conditions are both zeros. For example, $y_{-3} = y_{-2} = y_{-1} = y_0 = [0\ 0]'$, and $u_{-3} = u_{-2} = u_{-1} = [0\ 0]'$ for the first case. In the two examples, the closed-loop simulation is run over 200 sampling steps, and the desired references for $y_1$ and $y_2$ are randomly changed every 20 steps. Warm-start used in all solvers (qpOASES, OSQP, CDAL-ARX). We keep default solver settings in both qpOASES and OSQP, so that they produce solutions of similar precision, that is measured in terms of Euclidean distance (since qpOASES belongs to the class of active-set methods, in principle it always provides a high-precision solution at termination, so its solution quality cannot be tuned as easily as in the case of ADMM). For a fair comparison, in the two examples we set $\epsilon_{in} = 10^{-6}$ and $\epsilon_{out} = 10^{-6}$ under $\rho = 1$ to define the stopping criteria of our CDAL-ARX solver, so to obtain closed-loop control sequences with similar precision. In both examples, the generated closed-loop simulation results are almost indistinguishable, see Figures 1(a) and 1(b), respectively, which show good tracking performance and no violation in input and output constraints.

Using the qpOASES and OSQP solvers require the online construction of the QP problem, whose computation time must be counted in the total time. Table 1 lists the solution time of CDAL-ARX and lists the construction and solution time when using qpOASES (condensed construction) and OSQP (sparse construction). From Table 1 it can be noticed that CDAL-ARX is always solving the MPC problem in a smaller CPU time, when compared to the sum of construction and solution time of qpOASES and OSQP. Moreover, as the prediction horizon increases, qpOASES and OSQP may fail to solve the problem due to the ill-conditioning issue. Note also that the computation time of CDAL-ARX is often shorter than the pure solution time of qpOASES and OSQP (i.e., not counting the construction time), which seems to indicate that the reported speed-ups are due to both adopting the proposed augmented Lagrangian method and avoiding the the construction step.

## 5. CONCLUSION

This paper introduced a solution algorithm for solving MPC problems based on ARX models that avoids constructing the associated QP problem explicitly. Due to its




(a) Time-varying ARX model

(b) DNN-based LPV-ARX model

Fig. 1. Closed-loop tracking results

Table 1. Computation time (ms) of CDAL-ARX and comparison with other solvers

| Examples | T | CDAL-ARX avg, max | qpOASES avg, max | OSQP avg, max |
|---|---|---|---|---|
| Time-varying ARX | 10 | 0.14, 1.5 | 0.42*, 2.8*<br>0.08†, 1.4† | 0.41*, 2.9*<br>0.18†, 0.92† |
| | 20 | 0.25, 2.8 | 1.2*, 6.3*<br>0.18†, 4.4† | 1.0*, 3.8*<br>1.9†, 17† |
| | 30 | 0.36, 3.6 | 2.6*, 10.2*<br>fail | 2.4*, 8.1*<br>22†, 48† |
| DNN-based LPV-ARX | 10 | 0.51, 2.5 | 0.46*, 3.2*<br>0.57†, 3.9† | 0.42*, 3.6*<br>1.1†, 14† |
| | 20 | 1.2, 4.6 | 1.2*, 5.5*<br>fail | 0.97*, 4.5*<br>16†, 32† |
| | 30 | 2.0, 7.3 | 3.1*, 10.8*<br>fail | 2.8*, 8.9*<br>fail |

*construction time, †solution time. For qpOASES and OSQP the time to evaluate the MPC law is the sum of construction and solution time.

matrix-free and library-free features, the proposed CDAL-ARX algorithm can be useful in adaptive embedded linear MPC applications based on ARX models, especially when combined with a fast and robust recursive linear identification method. Future research will address extending the method to handle soft output constraints, so to relax Assumption 1.

## REFERENCES


Åström, K. and Wittenmark, B. (2013). *Adaptive control*. Courier Corporation.

Bertsekas, D. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

Huusom, J., Poulsen, N., Jørgensen, S., and Jørgensen, J. (2010). ARX-model based model predictive control with offset-free tracking. In *Computer Aided Chemical Engineering*, volume 28, 601–606. Elsevier.

Kang, M., Kang, M., and Jung, M. (2015). Inexact accelerated augmented Lagrangian methods. *Computational Optimization and Applications*, 62(2), 373–404.

Kouzoupis, D., Zanelli, A., Peyrl, H., and Ferreau, H. (2015). Towards proper assessment of QP algorithms for embedded model predictive control. In *2015 European Control Conference (ECC)*, 2609–2616. IEEE.

Luo, Z. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1), 7–35.

Mayne, D. (2014). Model predictive control: Recent developments and future promise. *Automatica*, 50(12), 2967–2986.

Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. akad. nauk Sssr*, volume 269, 543–547.

Qin, S. and Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11(7), 733–764.

Saraf, N. and Bemporad, A. (2017). Fast model predictive control based on linear input/output models and bounded-variable least squares. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1919–1924. IEEE.

Serra, T., Tjandraatmadja, C., and Ramalingam, S. (2018). Bounding and counting linear regions of deep neural networks. In *International Conference on Machine Learning*, 4558–4566. PMLR.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4), 637–672.

Wu, L. (2022a). Equivalence of SS-based MPC and ARX-based MPC. *arXiv preprint arXiv:2209.00107*.

Wu, L. (2022b). An interpretative and adaptive MPC for nonlinear systems. *arXiv preprint arXiv:2209.01513*.

Wu, L. and Bemporad, A. (2023). A Simple and Fast Coordinate-Descent Augmented-Lagrangian solver for Model Predictive Control. *IEEE Transactions on Automatic Control*, 1–8. doi:10.1109/TAC.2023.3241238.