Least Squares Support Vector Machines-based Imitation Learning of Nonlinear Model Predictive Control

Luca Cavanini[1,2], Francesco Ferracuti[1], Andrea Monteriù[1] and Francesco Vella[1]

***Abstract*— This paper presents the preliminary results of a Linear Parameter Varying-Autoregressive eXogenous model, identified through Least Squares Support Vector Machines, able to optimally drive a robotic arm system by emulating the performance of a Nonlinear Model Predictive Control (NMPC) policy. The support vector machine framework is employed to replicate the control performance of a computationally demanding NMPC. Due to the nonlinear characteristics of the robotic arm, the NMPC is suitable to guarantee expected control performance. However, its application in real-time systems with fast dynamics is limited by high memory and computational demands required at each sampling instant. In this work, the linear parameter varying model is trained using a data-driven approach to imitate the control actions of the NMPC across different scenarios. The proposed controller and the original NMPC are evaluated in simulation, considering multiple operating conditions of the robotic arm. The control performance of both approaches is then compared to assess the effectiveness of the proposed method.**

I. Introduction

In recent years, advances in control theory have enabled the application of the most advanced control paradigms to a wide range of systems, including those systems featured by strong nonlinear dynamics, constraints, and in presence of unpredictable disturbances and noise [1]. Among the various control strategies, nowadays, Model Predictive Control (MPC) is considered a feasible solution to achieve optimal performance from controlled plants [2]. MPC needs a mathematical model of the plant to predict the system dynamics over a specified time horizon, while optimizing the control result and directly taking into account physical and logical limits characterizing the controlled system [3]. When the plant is characterized by linear dynamics, the application of MPC to control problems characterized by fast dynamics represents a feasible approach. This is due to the form of the optimization problem to be solved that, in the case of linear-like systems, can be formulated as convex optimization problem that can be solved in real-time by exploiting several different optimization approaches requiring limited computational power and memory footprint [4]. This is commonly extended to quasi-linear systems by using different theoretical strategies, like the Linear Parameter-Varying (LPV) [5] or Linear Time-Varying (LTV) [6] modeling approaches or the iterative local linearization of the original nonlinear plant model [7].

Despite these advancements, in case of complex nonlinear plants, a suitable control strategy is the nonlinear extension of the predictive control framework, commonly termed Nonlinear Model Predictive Control (NMPC) [8]. NMPC is commonly formulated by using the nonlinear model of the plant, resulting in a non-convex optimization problem that requires higher computational resources to solve the optimization problem within the required sample time. In the optimization field, efficiently solving such problems remains an open challenge, which limits the application of NMPC in real-world control scenarios [9]. While various solutions have been successfully developed and implemented for linear or linear-like MPC formulations [10], [11], a general and effective approach has not been yet found for NMPC. In this paper, this limitation is addressed by leveraging the emulation capabilities of a Machine Learning (ML) technique, developed using a supervised learning approach. Although not a new concept in control research [12], several studies have proposed to use Artificial Intelligence (AI) to replicate the performance of complex control strategies that are otherwise unsuitable for real-time implementation due to their high computational demand.

In this work, a NMPC scheme designed to control a robotic arm is emulated by a Linear Parameter Varying-Autoregressive with eXogenous Input (LPV-ARX) model, identified through Least Squares Support Vector Machines (LS-SVM) [13]. LS-SVM is a modified version of SVM that employs an l_2 loss function, leading to a linear problem in an efficient computational way, providing a unique solution. This method has been applied to different data-driven identification problems, including LPV systems identification in both Single-Input Single-Output (SISO) and Multi-Input Multi-Output (MIMO) configurations, considering both transfer function and state-space form [14]. Nowadays, LS-SVM represents an effective and computationally efficient alternative to the most common Neural Network (NN) framework [15], and has been widely used in different control applications, due to its ability to capture complex functional relationships from data generated by unknown nonlinear processes [16]–[18]. In particular, the method adopted in this paper was originally proposed for nonlinear black-box system identification [19]. Here, the LPV-ARX model is trained to learn and reproduce the control actions of the NMPC tested over a wide set of operating scenarios during the control of a robotic manipulator's kinematic. After an initial training and calibration phase, the LPV-ARX model

[1]Luca Cavanini, Francesco Ferracuti, Andrea Monteriù and Francesco Vella are with the Department of Information Engineering, Università Politecnica delle Marche 60131 Ancona, Italy {l.cavanini, f.ferracuti, a.monteriu}@univpm.it, f.vella@pm.univpm.it

[2]Luca Cavanini is also with Industrial Systems and Control Ltd, Culzean House, 36 Renfield Street, Glasgow G2 1LU, UK l.cavanini@isc-ltd.com

is directly used to control, in closed-loop, the robot arm. Both baseline NMPC and LS-SVM-based LPV-ARX controller are tested and validated using a simulation model of the robotic arm, which is based on the nonlinear model of the 7-DOF Baxter manipulator [20], [21]. The two control strategies are compared in terms of control performance and computational efficiency to assess the effectiveness of the proposed approach.

This paper is organized as follows. Section II presents the mathematical model of the robot arm. Section III introduce the NMPC framework. Section IV describes the proposed LS-SVM-based approach. Section V provides the simulation performance and Section VI concludes the paper.

II. Robot Arm Model

In this section, the robotic arm system model used to test the proposed approach is presented. The robot is the 7-DOF Baxter manipulator. This is a redundant dual-arm manipulator, and here, the control of a single arm of the robot is considered. The Baxter arm consists of 7 revolute joints forming a serial manipulator and the mathematical model of the arm is derived using the Denavit–Hartenberg convention for forward kinematics. The robot, shown in Fig. 1, is characterized by the Denavit-Hartenberg (DH) parameters provided by the manufacturer, which are reported in Table I. The position and orientation of the end-effector relative to the base frame are derived using homogeneous transformation matrices defined by the DH parameters. The transformation matrix from frame $i-1$ to i is given by:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

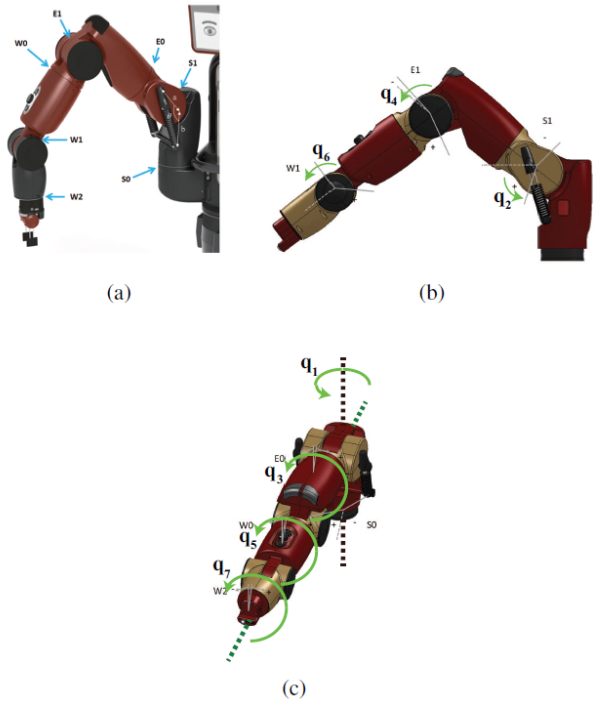
where θ_i is the i_{th} joint angle, d_i is the offset along previous z to the common normal, a_i is the length of the common normal and α_i is the angle about the common normal (from z_{i-1} to z_i). The complete transformation from the base to end-effector is obtained by the sequential multiplication of the individual transformations:

$$ {}^{0}T_7 = \prod_{i=1}^{7} {}^{i-1}T_i \tag{2} $$

where each ${}^{i-1}T_i$ is the homogeneous transformation matrix derived from the corresponding DH parameters. The kinematic model is directly derived from the well-known differential kinematics equation [22]:

$$ \dot{\mathbf{x}}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{3} $$

where $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^7$ represent the joint position and velocity vectors, respectively, $\dot{\mathbf{x}}_e \in \mathbb{R}^6$ is the end-effector velocity vector, containing both linear and angular velocity components, and $\mathbf{J}(\mathbf{q}) \in \mathbb{R}^{6\times 7}$ is the Jacobian Matrix of the robotic arm.



(a) (b) (c)

Fig. 1: (a) The 7-DOF Baxter manipulator; The joints' configuration: (b) sagittal view; (c) top view [23].

TABLE I: Baxter Manipulator Arm Specifications

Link	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]
1	0.069	0.27035	$-\pi/2$	θ_1
2	0	0	$\pi/2$	$\theta_2+\pi/2$
3	0.069	0.36435	$-\pi/2$	θ_3
4	0	0	$\pi/2$	θ_4
5	0.010	0.37429	$-\pi/2$	θ_5
6	0	0	$\pi/2$	θ_6
7	0	0.3945	0	θ_7

III. Nonlinear Model Predictive Control

MPC is a control policy able to exploit the full knowledge of the plant model to predict the behavior of the system over a predefined prediction horizon. One of its advantages is the ability to directly handle constraints characterizing the system dynamics. This is made by formulating and solving a constrained optimization problem. The solution of this problem provides the optimal control input at each sampling time. When the system to be controlled exhibits strong nonlinearities and several input, state and output variables, the optimization problem would become non-convex and computationally demanding to be solved within the limited sampling interval. Furthermore, increasing the prediction horizon, and introducing other advanced controller features (e.g., disturbance or reference previewing) would increase further the computational burden of the controller and limit its applicability in real-time control of systems with fast dynamics. In this work, a NMPC policy is employed for the kinematic control of the selected robot arm. The NMPC formulation considered in this paper is the following Eq. (4),

such that:

$$\min_{u} \quad \sum_{i=1}^{N_p-1} \|Q(x_{k+i|k} - x_{r_k})\|_2^2 + \sum_{j=0}^{N_u-1} \|R\delta u_{k+j|k}\|_2^2 \tag{4a}$$

$$\text{s.t.} \quad x_{k+i+1|k} = f(x_{k+i|k}, u_{k+i|k}) \tag{4b}$$

$$x_{k|k} = x_k \tag{4c}$$

$$u_{k+j|k} \in \mathbb{U} \tag{4d}$$

$$x_{k+i|k} \in \mathbb{X} \tag{4e}$$

$$i \in \{1, \dots, N_p - 1\} \tag{4f}$$

$$j \in \{0, \dots, N_u - 1\} \tag{4g}$$

where N_p is the prediction horizon, N_u is the control horizon, Q, R are the weight matrices penalizing the state tracking and the rate of change of the control effort, respectively. The term $x_{k+i|k}$ denotes the prediction of the variable x at time $k+i$, based on the information available at time k, $u_{k+i|k}$ is the vector of the input sequence, x_{r_k} is the state vector reference, $\mathbb{U}$ and $\mathbb{X}$ are sets of constraints on inputs and states, respectively. Eq.(4b) is the equality constraint of the nonlinear system model given by the nonlinear function $f(\cdot)$, corresponding to the plant dynamics introduced in Eq.(3), and reformulated in discrete-time as follows:

$$f : x(k+1) = x(k) + T_s \begin{bmatrix} \mathbf{J}(\mathbf{q}) \\ I \end{bmatrix} u(k) \tag{5}$$

with T_s the sampling time, I an identity matrix of appropriate dimension and considering following input and augmented state vectors:

$$u(k) = \dot{\mathbf{q}}(k) \ , \ x = \begin{bmatrix} \mathbf{x}_e(k) & \mathbf{q}(k) \end{bmatrix}' \tag{6}$$

with $\mathbf{x}_e(k)$ the end-effector pose and $\mathbf{q}(k)$ the joints state. This model is characterized by constraints embedded as in Eq. (4d) and Eq. (4e). The computational complexity, required to solve this optimization problem for general nonlinear systems, would be too high for implementation on real-time control hardware. To address this limitation, the aim of this work is to develop a ML model that can be executed on embedded boards while maintaining comparable performances to that of the NMPC. This is achieved by testing the NMPC on the simulated model of the robotic arm and storing optimal NMPC input and output signals such that a dataset is collected to identify the LPV-ARX model able to approximate the control actions of the NMPC. In the following section, tthe LS-SVM-based LPV-ARX modeling approach is introduced. The formulation of the control policy is presented, and the procedure for using the collected dataset to emulate the behavior of the NMPC is described.

IV. Least Squares Support Vector Machines

In this section, the LS-SVM framework for the identification of LPV-ARX models is presented. This machine learning approach is employed to develop an algorithm capable of effectively replicating the performance of the NMPC introduced in the previous section, while significantly reducing the associated computational complexity. A SISO discrete-time LPV system in ARX form can be expressed as:

$$\begin{aligned} y(k) + \sum_{i=1}^{n_a} a_i(\boldsymbol{\rho}(k)) y(k-i) = \\ \sum_{i=1}^{n_b} b_{i-1}(\boldsymbol{\rho}(k)) u(k-i+1) + e(k) \end{aligned} \tag{7}$$

where $k \in \mathbb{Z}$ is the discrete time index, n_a is the output order, n_b is the input order, $u : \mathbb{Z} \to \mathbb{U} \subseteq \mathbb{R}$ is the vector of input variables, $y : \mathbb{Z} \to \mathbb{Y} \subseteq \mathbb{R}$ is the vector of output variables, $\boldsymbol{\rho} : \mathbb{Z} \to \mathbb{P} \subseteq \mathbb{R}^{n_p}$ is the vector of scheduling variables and $e : \mathbb{Z} \to \mathbb{E} \subseteq \mathbb{R}$ is a vector of white stochastic noise processes. The time-varying parameters $a_i(\boldsymbol{\rho}) : \mathbb{P} \to \mathbb{R}$ and $b_i(\boldsymbol{\rho}) : \mathbb{P} \to \mathbb{R}$ have a static dependence on the scheduling parameter vector $\boldsymbol{\rho}$. Without loss of generality and to simplify the notation, in the proposed approach only the instantaneous value of the scheduling parameters vector $\boldsymbol{\rho}(k)$ influences the value of a_i and b_i. The number of parameters to be estimated is $n_g = n_a + n_b$. A usual assumption in LPV system identification is that the coefficients a_i, b_i are linear combinations of a set of basis functions:

$$\begin{aligned} a_i(\boldsymbol{\rho}(k)) &= \sum_{\nu=1}^{n_H} \omega_{i,\nu} \phi_{i,\nu}(\boldsymbol{\rho}(k)) = \boldsymbol{\omega}_i^T \boldsymbol{\phi}_i(\rho(k)), \ i = 1, \dots, n_a \\ b_i(\boldsymbol{\rho}(k)) &= \sum_{\nu=1}^{n_H} \omega_{\tilde{i},\nu} \phi_{\tilde{i},\nu}(\boldsymbol{\rho}(k)) = \boldsymbol{\omega}_{\tilde{i}}^T \boldsymbol{\phi}_{\tilde{i}}(\boldsymbol{\rho}(k)), \ i = 1, \dots, n_b \end{aligned} \tag{8}$$

where $\tilde{i} = n_a + 1, \dots, n_g$, $\boldsymbol{\phi}_i(\boldsymbol{\rho}(k)) : \mathbb{R}^{n_p} \to \mathbb{R}^{n_H}$ denotes an undefined, potentially infinite dimensional ($n_H = \infty$) feature map with static dependence on $\boldsymbol{\rho}(k)$, and $\boldsymbol{\omega}_i \in \mathbb{R}^{n_H}$ is a vector of weights. By using (8), LPV-ARX model can be rewritten in regression form:

$$\hat{y}(k) = \sum_{i=1}^{n_g} \boldsymbol{\omega}_i^T \boldsymbol{\phi}_i(\boldsymbol{\rho}(k)) x_i(k) \tag{9}$$

The goal of the identification process in primal space is to find an approximation of the feature maps $\boldsymbol{\phi}_i$ and subsequently calculate the weight vectors $\boldsymbol{\omega}_i$ given the identification dataset $\mathcal{D}_N = \{u(k), \rho(k), y(k)\}_{k=1}^N$, where N is the number of observations and

$$x_i(k) = \begin{cases} y(k-i) & \text{if } i = 1, \dots, n_a \\ u(k-i+n_a+1) & \text{if } i = n_a+1, \dots, n_g \end{cases} \tag{10}$$

In this work, we consider the Multi-Input Multi-Output (MIMO) version of the model and the dual form defined as follows. Defining the vector $\mathbf{Y} = \begin{bmatrix} y(1), \dots, y(N) \end{bmatrix}^T$, the regression equation can be written in compact form

$$\mathbf{Y} = \left(\mathbf{K} + \gamma^{-1} I_N \right) \boldsymbol{\alpha} = \boldsymbol{\Omega} \boldsymbol{\alpha} \tag{11}$$

where I_N is the identity matrix of dimension N, γ is the regularization parameter of the LS-SVM problem and the

matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ is computed as

$$K(\boldsymbol{\rho}(k), \boldsymbol{\rho}(\tilde{k})) = \sum_{i=1}^{n_g} x_i(k) \underbrace{\boldsymbol{\phi}_i(\boldsymbol{\rho}(k))^T \boldsymbol{\phi}_i(\boldsymbol{\rho}(\tilde{k}))}_{K_i^g(\boldsymbol{\rho}(k),\boldsymbol{\rho}(\tilde{k}))} x_i(\tilde{k}) \tag{12}$$

where k is related to the training dataset, and $\tilde{k}$ to the prediction dataset. Rearranging Eq. (12), the matrix $\mathbf{K} \in \mathbb{R}^{N\times N}$ can be defined as the Hadamard product of two kernel matrices as

$$\mathbf{K} = \sum_{i=1}^{n_g} \mathbf{K}_i^g \circ \mathbf{K}_i^l \tag{13}$$

where $\mathbf{K}_i^g$ is a kernel matrix whose kernel function is defined *a priori*, whereas $\mathbf{K}_i^l$ is the kernel matrix of a linear kernel function. Considering the Radial Basis Function (RBF) as kernel function, therefore the entries of $\mathbf{K}_i^g$ are

$$K_i^g(\boldsymbol{\rho}(k), \boldsymbol{\rho}(\tilde{k})) = \exp\left(-\frac{\|\boldsymbol{\rho}(k) - \boldsymbol{\rho}(\tilde{k})\|_2^2}{2\sigma_i}\right) \tag{14}$$

where σ_i are the kernel parameters. The matrix $\mathbf{K}$ is used to compute the vector of Lagrangian multipliers $\boldsymbol{\alpha}$ representing the solution of Eq. (11)

$$\boldsymbol{\alpha} = \boldsymbol{\Omega}^{-1}\mathbf{Y} \tag{15}$$

and the solution shown in Eq. (15) of LS-SVM dual formulation corresponds to a form of ridge regression. Based on the kernel trick, the coefficients $a_i(\boldsymbol{\rho}(\tilde{k}))$ and $b_i(\boldsymbol{\rho}(\tilde{k}))$ are estimated without explicitly defining the involved feature maps $\boldsymbol{\phi}_i$. This gives that the estimated coefficient functions are obtained as

$$\hat{a}_i(\boldsymbol{\rho}(\tilde{k})) = \sum_{k=1}^{N} \alpha(k) x_i(k) \mathbf{K}_i^g(\boldsymbol{\rho}(k), \boldsymbol{\rho}(\tilde{k})), \quad i = 1, \ldots, n_a \tag{16}$$

$$\hat{b}_i(\boldsymbol{\rho}(\tilde{k})) = \sum_{k=1}^{N} \alpha(k) x_{\tilde{i}}(k) \mathbf{K}_i^g(\boldsymbol{\rho}(k), \boldsymbol{\rho}(\tilde{k})), \quad i = 1, \ldots, n_b \tag{17}$$

V. Simulation results

The task addressed in this work is the kinematic control of the Baxter robotic arm to simulate a handwriting motion—specifically, tracking a desired end-effector pose trajectory. The reference trajectory dataset is a pre-processed version of the 'Simplex' dataset originally introduced by Hershey [24] and later adapted by Corke [25]. The dataset, sampled at 10 Hz, provides Cartesian coordinates representing the first 128 ASCII characters. From these, 41 usable trajectories were selected, by discarding short trajectories having less than 10 samples. To test the baseline NMPC controller, each of the selected trajectories was executed 10 times, with white Gaussian noise $\mathcal{N}(0, 0.05)$ added to the plant input signals. This approach increased the variability of the data, enhancing the robustness of the LPV-ARX model identification across a wider range of operating conditions. For each trajectory $i-$th, the corresponding dataset is denoted as $D_i = \{\bar{u}_k, \rho_k, \bar{y}_k\}$. The LPV-ARX systems input is $\bar{u}_k = [x(k), x_r(k)]'$, with $x(k)$ the robot states and $x_r(k)$ the robot state reference signals. The scheduling variable $\rho_k = u(k-1)$ is the previous NMPC control move and $\bar{y}_k = u(k)$ is the computed control action driving the robot. Noisy trajectories were used for training the LS-SVM models, while noise-free reference trajectories were employed for the testing of the proposed ML. The NMPC tuning parameters are reported in Table II, while the LS-SVM hyparameters σ and γ were optimized separately for each symbol and its associated set of trajectories using a grid search based on the Best-Fit-Rate performance metric. The considered work has been implemented within the Mathworks MATLAB/Simulink suite by using the $fmincon$ routine to solve the NMPC optimization problem.

TABLE II: NMPC Calibration Parameters

Parameter	Value
T_s	0.1 [s]
N_p	10
N_u	5
R	$I \times 0.1$
Q	I
u^m	$-[1.5, 1.5, 1.5, 1.5, 4, 4, 4]$ [rad/s]
u^M	$[1.5, 1.5, 1.5, 1.5, 4, 4, 4]$ [rad/s]
x^m	$-[1.702, 1.047, 3.054, 2.618, 3.059, 2.094, 3.059]$ [rad]
x^M	$[1.702, 1.047, 3.054, 2.618, 3.059, 2.094, 3.059]$ [rad]

Table III summarized the control performances of both the NMPC and the LS-SVM-based controller, averaged over the entire set of nominal, noise-free trajectories. This table reports the Root Mean Square Error (RMSE) for both linear and angular end-effector pose, including the mean values and related standard deviations. Additionally, the maximum linear position error is provided, along with the control policy computational time.

TABLE III: Results evaluated on the tracking of 41 trajectories

	NMPC	LPV-ARX
RMSE Linear [cm]	0.21± 0.09	0.85± 0.25
RMSE Angular [rad]	0.0008± 0.0003	0.007± 0.0092
Max Linear Error [cm]	0.88± 0.4	1.91± 0.65
Inference Time [ms]	224.6± 310	0.27± 0.26

The results show that the proposed approach, while less accurate than the NMPC in terms of tracking performance accuracy, reduces computational complexity and execution time. This trade-off is the key outcome of the study, given that the NMPC requires an average computation time of 225 ms, which exceeds the sampling time constraint of 0.1 s, making real-time implementation infeasible.

Fig. 2 and Fig. 3 show the trajectory tracking performances for symbols 1 and 19, respectively. Furthermore, Fig. 4, Fig. 5 and Fig. 6 show the control performances along the three axis for symbol 1 and Fig. 7, Fig. 8 and Fig. 9 for symbol 19, respectively. These results show that the proposed ML achieves results comparable to NMPC, while reducing the computational burden.

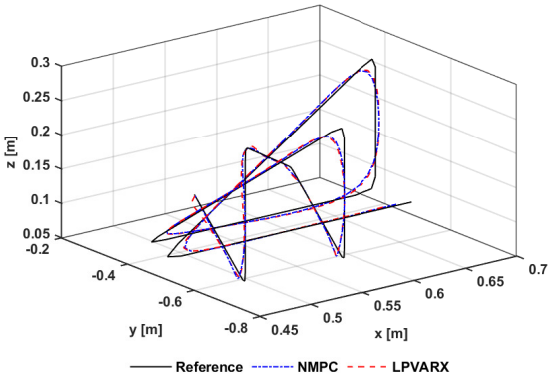


Fig. 2: Trajectories of symbol 1

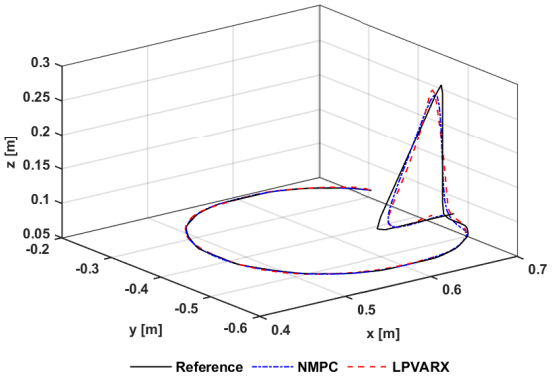


Fig. 3: Trajectories of symbol 19

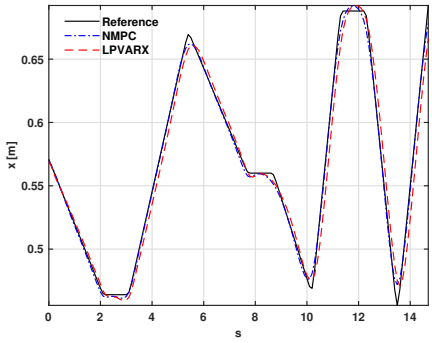


Fig. 4: Trajectories of symbol 1 along x-axis

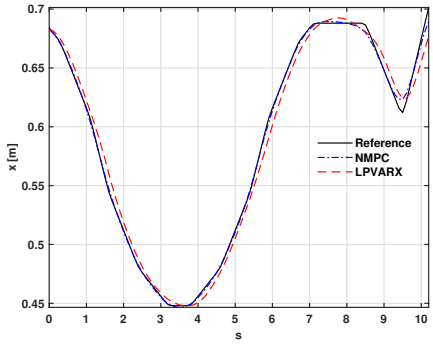


Fig. 5: Trajectories of symbol 19 along x-axis

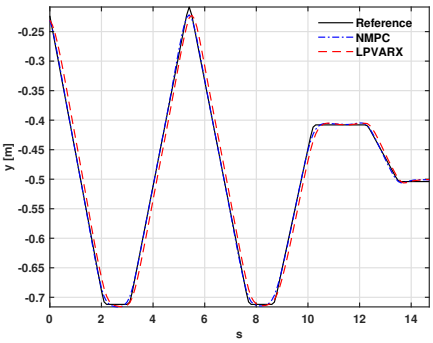


Fig. 6: Trajectories of symbol 1 along y-axis

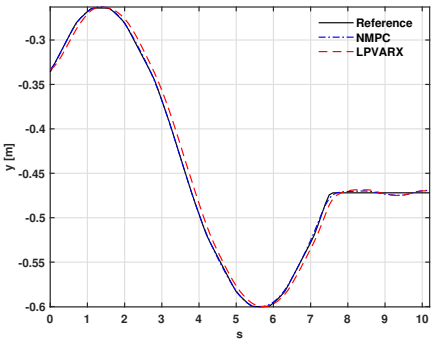


Fig. 7: Trajectories of symbol 19 along y-axis

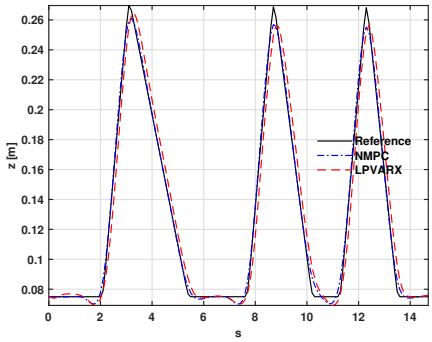


Fig. 8: Trajectories of symbol 1 along z-axis

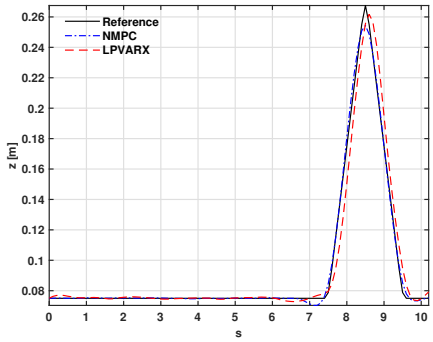


Fig. 9: Trajectories of symbol 19 along z-axis

VI. Conclusions

In this paper, a control approach based on Linear Parameter-Varying Autoregressive with eXogenous Input (LPV-ARX) model identified using a Least Squares Support Vector Machines (LS-SVM) framework has been proposed, to mimic the control performance of a Nonlinear Model Predictive Control (NMPC) policy designed to drive a robotic arm. The NMPC was designed to control a simulated model of the robotic arm, leveraging its ability to deliver optimal performance for complex nonlinear systems subject to physical and logical constraints. However, due to the high computational complexity and memory requirements associated with solving the NMPC optimization problem in real time, direct implementation on embedded control hardware is not feasible. To address this limitation, an LS-SVM-based LPV-ARX model has been trained to emulate the baseline NPMC by using data collected from multiple NMPC-controlled simulation runs. The trained model was then validated in simulation and its performance was compared to the baseline NMPC controller. The results demonstrate that the LS-SVM-based controller achieves comparable control accuracy while significantly reducing computational time. Future research will be focused to the test of the developed LS-SVM-based LPV-ARX controller on a real robotic arm. This will include the analysis of the computational complexity of the LS-SVM-based LPV-ARX model, in order to optimally evaluate the trade-off between performance and algorithm execution effort. To further reduce the computational burden and memory footprint, especially when trained on large datasets, techniques for dataset selection and pruning—under both open-loop and closed-loop configurations—will be investigated. These enhancements aim to enable real-time execution of the proposed LS-SVM-based controller on low-power embedded systems, while maintaining the desired control performance.

References

[1] T. Samad, M. Bauer, S. Bortoff, S. Di Cairano, L. Fagiano, P. F. Odgaard, R. R. Rhinehart, R. Sánchez-Peña, A. Serbezov, F. Ankersen, et al., "Industry engagement with control research: Perspective and messages," Annual Reviews in Control, vol. 49, pp. 1–14, 2020.

[2] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," The International Journal of Advanced Manufacturing Technology, vol. 117, no. 5, pp. 1327–1349, 2021.

[3] F. Borrelli, A. Bemporad, and M. Morari, Predictive control for linear and hybrid systems. Cambridge University Press, 2017.

[4] J. A. Rossiter, Model-based predictive control: a practical approach. CRC press, 2017.

[5] L. Cavanini, G. Ippoliti, and E. F. Camacho, "Model predictive control for a linear parameter varying model of an uav," Journal of Intelligent & Robotic Systems, vol. 101, no. 3, p. 57, 2021.

[6] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation," International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal, vol. 18, no. 8, pp. 862–875, 2008.

[7] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," International Journal of Control, vol. 93, no. 1, pp. 62–80, 2020.

[8] L. Grüne, J. Pannek, L. Grüne, and J. Pannek, Nonlinear model predictive control. Springer, 2017.

[9] G. Cimini and A. Bemporad, "Exact complexity certification of active-set methods for quadratic programming," IEEE Transactions on Automatic Control, vol. 62, no. 12, pp. 6094–6109, 2017.

[10] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," Nonlinear Model Predictive Control: Towards New Challenging Applications, pp. 345–369, 2009.

[11] L. Cavanini, G. Cimini, and G. Ippoliti, "Computationally efficient model predictive control for a class of linear parameter-varying systems," IET Control Theory & Applications, vol. 12, no. 10, pp. 1384–1392, 2018.

[12] A. Tagliabue and J. P. How, "Efficient deep learning of robust policies from mpc using imitation and tube-guided data augmentation," IEEE Transactions on Robotics, 2024.

[13] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," Neural processing letters, vol. 9, pp. 293–300, 1999.

[14] L. Cavanini, R. Felicetti, F. Ferracuti, and A. Monteriù, "Fixed-size ls-svm lpv system identification for large datasets," International Journal of Control, Automation and Systems, vol. 21, no. 12, pp. 4067–4079, 2023.

[15] V. Kecman, Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. MIT press, 2001.

[16] L. Cavanini, P. Majecki, M. J. Grimble, and G. M. van der Molen, "Battery state-of-charge estimator design based on the least-squares support vector machine," in 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE). IEEE, 2024, pp. 711–716.

[17] L. Cavanini, F. Ferracuti, S. Longhi, E. Marchegiani, and A. Monteriù, "Sparse approximation of ls-svm for lpv-arx model identification: Application to a powertrain subsystem," in 2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE). IEEE, 2020, pp. 1–6.

[18] L. Cavanini, F. Ferracuti, S. Longhi, and A. Monteriù, "Ls-svm for lpv-arx identification: Efficient online update by low-rank matrix approximation," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), 2020, pp. 1590–1595.

[19] H. Wang and D. Hu, "Comparison of svm and ls-svm for regression," in 2005 International conference on neural networks and brain, vol. 1. IEEE, 2005, pp. 279–283.

[20] A. Smith, C. Yang, C. Li, H. Ma, and L. Zhao, "Development of a dynamics model for the baxter robot," in 2016 IEEE international conference on mechatronics and automation. IEEE, 2016, pp. 1244–1249.

[21] A. Freddi, S. Iarlori, S. Longhi, and A. Monteriù, "Development and experimental validation of algorithms for human–robot interaction in simulated and real scenarios," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 4, pp. 4529–4539, 2021.

[22] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, Robotics: Modelling, Planning and Control. Springer, 2010.

[23] M. Bagheri, M. Krstić, and P. Naseradinmousavi, "Analytical and experimental predictor-based time delay control of baxter robot," in Dynamic Systems and Control Conference, vol. 51890. American Society of Mechanical Engineers, 2018, p. V001T04A011.

[24] P. Bourke, "Hershey Vector Font repository," https://paulbourke.net/dataformats/hershey/.

[25] P. Corke, W. Jachimczyk, and R. Pillat, Robot Arm Kinematics. Cham: Springer International Publishing, 2023, pp. 275–328.