2025 American Control Conference (ACC)
July 8-10, 2025. Denver, CO, USA

Data-Driven Model Predictive Control of an Hydraulic Excavator via Local Model Networks

Salim Msaad, Leonardo Cecchin, Ozan Demir, Lorenzo Fagiano

***Abstract*— A novel solution to control an hydraulic excavator during grading tasks is proposed, featuring a Model Predictive Controller designed using Local Model Networks (LMNs), i.e. linear time-invariant dynamic models averaged by nonlinear static functions. The Local Linear Models Tree (LoLiMoT) algorithm is employed to derive an LMN from experimental data of a real excavator. Then, a nonlinear MPC law is designed and implemented on the excavator's embedded control system. To further improve the computational efficiency, a time-varying MPC law is designed as well, where the LMN is linearized in real-time around the current operating point. Experimental results, conducted with the excavator in real-world conditions, show the effectiveness of both approaches in achieving performance comparable to state-of-the-art solutions, while utilizing a more compact dataset and without the need of the hydraulic cylinders' pressure measurement.**

I. Introduction

Assistance functions for earth moving machines, like excavators, currently represent a research area of significant interest. These functions serve to support operators over repetitive and monotonous tasks, enhancing overall productivity.

Focusing on excavators, grading stands out as one of the most time consuming and repetitive tasks encountered. It is the process of shaping the ground surface to attain a specific slope. For this task, the excavator arm, equipped with a specialized grading bucket, needs to be controlled precisely along a predefined path. There is a wide variability of efficiency and performance in grading operations; the remarkable ability of the best professional operators in doing so sets high requirements for assistance functions to take over the task. This, combined with the highly nonlinear behavior of the hydraulic systems, impedes the use of simple control structures. Model-based methods provide a way to achieve these requirements, however, their implementation is often deemed impractical, due to the diverse range of excavator models, each manufactured in limited quantities. As a result, approaches leveraging learning from data are preferred.

Various data-driven methodologies in the application of hydraulic excavators are documented in the literature. In [1], reinforcement learning is used to obtain a control policy without the need of machine-specific knowledge. Deep convolutional neural networks are proven to be effective in [2], when used in feed-forward control strategies to control heavy-duty hydraulic manipulators. In [3], the authors propose a data-driven PID control scheme to deal with the nonlinearities that characterize the hydraulic system.

In addition to exclusively data-driven methods, the literature also features hybrid approaches that combine knowledge from the physical system with information extracted from collected data. In [4], the authors present a hybrid approach where neural networks and simple physical models are used for inverse model control. This hybrid approach is then compared against a purely data-driven controller and a physics based controller, comparing performance, engineering effort and safety. In [5], a similar approach is considered together with gaussian process models to capture the inverse system behaviour, incorporating prior knowledge and considerations of uncertainty. Building on this, [6] explores safe active learning for internal model control, which has demonstrated effectiveness in minimizing the amount of data required.

Model predictive control (MPC) [7] also appears in studies concerned with the control of hydraulic excavators and other hydraulic machinery, thanks to its capacity to optimize performance while considering the system's constraints. In [8], the authors explore the applicability of this control methodology in the grading process of an excavator. They develop a nonlinear model based on physical knowledge of the system and implement it in a nonlinear MPC, demonstrating precise and fast tracking in simulation. Another application of MPC to hydraulic excavators is presented in [9], where offset-free model predictive control is evaluated on a mini excavator.

Differently from the mentioned literature, in this study we explore a data-driven MPC design for excavator digging assistance functions, where the predictive model is obtained from experimental data using a black-box approach. Various model architectures can be considered for this purpose; here, Local Model Networks (LMNs) [10] are employed. This choice is motivated by their ability to model nonlinear dynamics with a modular structure, which allows for efficient linearization. The Local Linear Model Tree (LoLiMoT) algorithm is used to derive the LMN from data [11]. The combined use of LMNs and MPC is seldom encountered in the literature, to the best of our knowledge; moreover, two MPC laws are here designed, implemented, and tested on a real excavator: a nonlinear MPC law, and a linear time-varying (LTV) one, where the LMN is linearized in real-time around the current operating point. Finally, the designed MPC laws are also compared with a reference controller from [4], showing the superiority of the proposed method.

*This project has received funding from the European Union's Framework Programme for Research and Innovation Horizon 2020 (2014-2020) under the Marie Sklodowska-Curie Grant Agreement No. 953348

Salim Msaad is with the Delft Center for Systems and Control (DCSC), Faculty of Mechanical Engineering, Delft University of Technology, Netherlands `s.msaad@tudelft.nl`

Leonardo Cecchin and Lorenzo Fagiano are with the Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Italy. `leonardo.cecchin@polimi.it`, `lorenzo.fagiano@polimi.it`

Ozan Demir is with Corporate Research of Robert Bosch GmbH, Renningen, Germany. `ozan.demir@de.bosch.com`

II. System description and control task

For the task at hand, we focus on the arm assembly of the excavator. It can be divided into two interacting sub-systems: the mechanical and the hydraulic one. The mechanical sub-system is composed of a fixed body, the *uppercarriage*, and three movable links: *boom*, *arm* and *bucket*, as shown in Fig. 1. These links are connected via three rotating joints, that are in turn actuated via hydraulic cylinders. The machine has a split-boom configuration, in which an additional cylinder - *tab* - can be used to change the shape of the boom. This additional degree of freedom is not used in the grading task, so it is kept constant and neglected in the remainder.

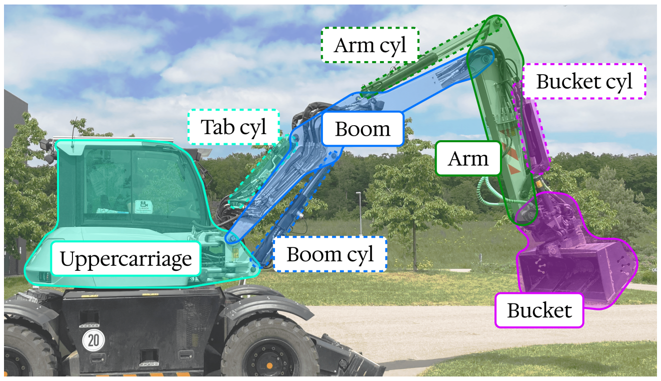

Fig. 1: Test vehicle under exam. Areas with a continuous perimeter represent the three links, while areas with a dotted perimeter represent the actuators.

The cylinders move thanks to the action of pressurized hydraulic fluid, which is provided by a variable displacement pump, and controlled through a set of continuous valves. The pump is in turn powered by an internal combustion engine. At each discrete time instant t, a control signal $j_{(\cdot)}(t)$ is translated into a reference speed for the corresponding joint, $v_{(\cdot)}(t)$. The input and output vectors of the system, $j(t)$ and $v(t)$ respectively, are therefore

$$\begin{aligned} j(t) &= [j_{\text{boom}}(t), j_{\text{arm}}(t), j_{\text{bucket}}(t)]^\top \in \mathbb{R}^3, \\ v(t) &= [v_{\text{boom}}(t), v_{\text{arm}}(t), v_{\text{bucket}}(t)]^\top \in \mathbb{R}^3. \end{aligned} \tag{1}$$

Regarding the considered control task, Fig. 2 illustrates the grading process, showcasing how the tool center point (TCP) of the bucket needs to describe a predetermined line, while the bucket is kept at a constant inclination. The control problem consists in simultaneously controlling the input signals such that the TCP follows the desired path. When the excavator is operated manually, a joystick is used to issue all components of vector $j(t)$. We provide next an overview of the chosen modeling framework, followed by a description of the specific data and identification procedure carried out in this work.

III. Local Model Networks

A. Model Architecture

LMNs are structured as networks where each neuron consists of a Local Linear Model (LLM) and a weighting function that practically delimits its region of validity, [10]. The architecture of a dynamic LMN is represented in Fig. 3.

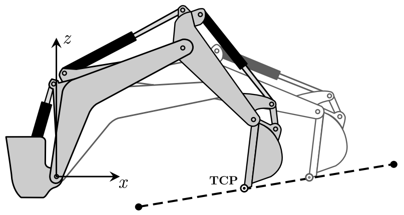

Fig. 2: Graphical illustration of the grading process. The TCP moves along a straight line in the working space while the bucket angle is kept constant.

The linear models can be written in ARX form: denoting with $u(t) \in \mathbb{R}^l$, $y(t) \in \mathbb{R}$ the input and output, we build the regressor $\varphi(t) \in \mathbb{R}^p$ as:

$$\begin{aligned} \varphi(t) = [&u(t-1)^\top, \dots, u(t-m_u)^\top, \\ &y(t-1), \dots, y(t-m_y)]^\top. \end{aligned} \tag{2}$$

With m_u, m_y being positive integers and $\cdot^\top$ the transpose operation.

The output of the i-th local linear model is given by $y_i(t) = w_i\,\varphi(t) + \xi_i$. where $w_i \in \mathbb{R}^p$ and $\xi_i \in \mathbb{R}$ represent the linear parameters of the LLM.

On the other hand, the validity functions, $\Phi_i(\varphi)$, are employed to introduce a regressor-dependent weighted average of the trajectories of the LLMs, such that, for a network with M neurons, we have $\sum_{i=1}^M \Phi_i(\varphi) = 1$. By introducing the membership function $\mu_i(\varphi)$ of each neuron as an axis-orthogonal Gaussian function with center $c_i = [c_{i1}, \dots, c_{ip}]^\top \in \mathbb{R}^p$ and standard deviation $\sigma_i = [\sigma_{i1}, \dots, \sigma_{ip}]^\top \in \mathbb{R}^p$, i.e.

$$\mu_i(\varphi) = \exp\left(-\frac{1}{2}\left(\frac{(\varphi_1 - c_{i1})^2}{\sigma_{i1}^2} + \dots + \frac{(\varphi_p - c_{ip})^2}{\sigma_{ip}^2}\right)\right),$$

the respective validity function is defined as:

$$\Phi_i(\varphi) = \frac{\mu_i(\varphi)}{\sum_{j=1}^M \mu_j(\varphi)}. \tag{3}$$

The values of c_i, σ_i and w_i, for each $i = 1, \dots, M$, are to be determined in the training phase of the model. The output of the network is finally obtained as the weighted sum of the outputs of all the LLMs:

$$y(t) = \sum_{i=1}^M \Phi_i(\varphi(t))(w_i\,\varphi(t) + \xi_i). \tag{4}$$

B. Training procedure

Identifying an LMN from data means determining the coefficients of each local linear model, together with the centers and standard deviations of each validity function, such that an error metric with respect to the training data-set is minimized.

If the validity functions are known, estimating the linear parameters of the LLMs is a linear least-squares problem,

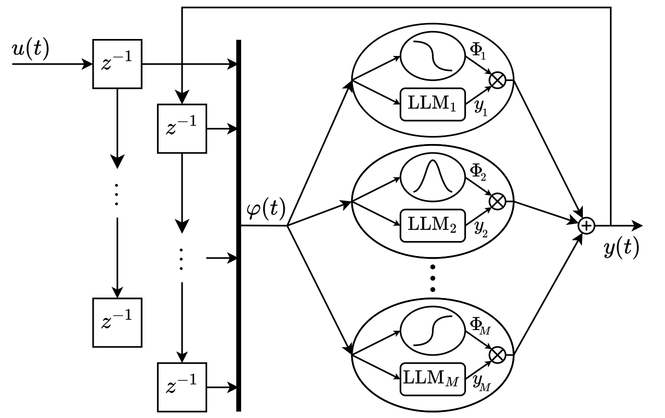

Fig. 3: Structure of a dynamic LMN with M LLMs. Here, the z^{-1} operator indicates a one time step delay.

whose global minimizer can be computed in close form provided that the available data is informative enough.

Given a set of N_d measured inputs $\{\tilde{u}(1),\dots,\tilde{u}(N_d)\}$ and outputs $\{\tilde{y}(1),\dots,\tilde{y}(N_d)\}$, it is possible to construct the vector $\tilde{\varphi}(t)\in\mathbb{R}^p$ for each measurement as in Eq. (2). As described in [10], the regression matrix $X\in\mathbb{R}^{N_d\times(p+1)}$ can be defined as $X=[\mathbf{1}_{N_d},\,\tilde{\boldsymbol{\varphi}}]$, where $\mathbf{1}_{N_d}$ is an all-ones column vector of length N_d and $\tilde{\boldsymbol{\varphi}}=[\tilde{\varphi}(1),\dots,\tilde{\varphi}(N_d)]^\top$. The vector $y_i=[y_i(1),\dots,y_i(N_d)]^\top$, that collects the outputs of the i-th model for each data point, is then given by $y_i=X\overline{w}_i$, where $\overline{w}_i=[\xi_i,w_i]^\top$ is the parameter vector.

The least squares problem that has to be solved for each model is:

$$\min_{\overline{w}_i}\sum_{j=1}^{N_d}\Phi_i\left(\tilde{\varphi}(j)\right)\left(X\overline{w}_i-\tilde{y}(j)\right)^2. \tag{5}$$

By defining the weighting matrix $\mathcal{Q}_i\in\mathbb{R}^{N_d\times N_d}$ as:

$$\mathcal{Q}_i=\begin{bmatrix}\Phi_i(\tilde{\varphi}(1)) & 0 & \cdots & 0\\ 0 & \Phi_i(\tilde{\varphi}(2)) & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \Phi_i(\tilde{\varphi}(N_d))\end{bmatrix} \tag{6}$$

the solution of the least squares problem is given by:

$$\tilde{w}_i^*=\left(X^\top\mathcal{Q}_iX\right)^{-1}X^\top\mathcal{Q}_i\,y_i. \tag{7}$$

C. LoLiMoT Algorithm

The linear optimization problem used to estimate the linear parameters presented in the previous subsection is based on the assumption that each validity function is known. The validity functions determine a partition of the input space, thus, the actual assumption that has been made is that the partitioning of the input space is known. There are different possibilities on how this partitioning task can be tackled, the Local Linear Model Tree (LoLiMoT) algorithm proposed in [10] offers a possible data-driven procedure based on axis-orthogonal partitioning.

At each iteration the algorithm splits the worst partition, creating two neurons from the one with the lowest performance, the linear parameters of LLMs for these newly added regions are then computed in an inner loop. The LoLiMoT algorithm thus features the following iterative steps:

1) *Start with an initial model:* Compute the bounds of the input space based on the data samples and construct the initial membership function of the single neuron of the LMN.
2) *Find the worst LLM:* Following a local estimation approach, the worst neuron is found as:

$$\arg\max_i\sum_{j=1}^{N}\Phi_i(\varphi(j))\left(y(j)-\tilde{y}(j)\right)^2. \tag{8}$$

3) *Test all possible subdivisions:* Only the worst LLM is considered for further refinement. Two halves are obtained by splitting the hyperrectangle of this LLM with an axis-orthogonal split. Divisions in all p dimensions are tried, computing all of the validity functions and linear parameters. The center of each validity function is determined as the center of its relative hyperrectangle, while the standard deviation of the i-th model along the j-th dimension, i.e. σ_{ij}, is one third of the length of the hyperrectangle along this dimension. The linear parameters are computed solving the least squares problem described in Section III-B.
4) *Find the best subdivision:* The overall model error for each alternative subdivision is computed. The best of the p alternatives is selected. The validity functions and the LLMs previously computed in step 3 are adopted.
5) *Test for convergence:* If the termination criterion is met then stop, else go to step 2. Various termination criteria are applicable in this step, such as setting a maximum model complexity, defined by a maximum number of LLMs, conducting statistical validation tests, or using more sophisticated cross-validation strategies.

IV. Model Identification

In this section we illustrate how the LMN architecture and the LoLiMoT algorithm, explained previously, are applied to obtain a model for our system.

A. Dataset Collection

The dataset used to train the LMN is obtained through experiments where amplitude-modulated Pseudo-Random Bit Sequence (APRBS) signals [12] and multi-sines were used as excitation signals, similar to the ones carried out by the authors in [4]. During the data collection, the excitation signals are generated such that the boom and the arm input signals cover their full range, while for the bucket a smaller range of $\pm70\%$ of the maximum amplitude is considered more appropriate.

These experiments have been carried out through movements in the air, without any load. A safety function is needed to prevent ground contact during the data collection. Without it, there is a high possibility that the system would impact the ground, degrading the quality of the data and possibly damaging the working surface. This safety function measures the TCP height and switches the sign of the current input signals every time it crosses a certain threshold.

Vel. Model	Boom	Arm	Bucket
Training Time [s]	3.0	2.8	4.7
Number of LLMs	20	20	28
RMSE [m/s]	0.0059	0.0086	0.0048

TABLE I: Training results of the LMNs for the velocity model of each hydraulic cylinder.

Additionally, this dataset includes measurements from leveling tasks conducted using a separate controller, contributing 40% of the total training data.

The recorded data contains the input signals and the resulting velocities for the three cylinders, $j^d(t)$ and $v^d(t)$ respectively. The velocity signals are obtained through forward finite differences applied to the measured cylinder positions. This process introduces significant numerical noise that is partially mitigated through the use of a first order filter.

Approximately 72 minutes of data were collected with a sampling time of 10ms. This data is then split with a ratio of 70% for the training dataset, 20% for the validation dataset and 10% for the testing dataset. In the remainder, we refer to this dataset as `Dataset-A`.

B. Model Training

We trained a model that captures the relationship between the input signals $u(t) = j(t)$ and the velocities of the cylinders $y(t) = v(t)$. To obtain this model, we trained three LMNs (one for each cylinder) that are then used together, such that, when used in simulation, the regressor of each model is composed of the velocities and input signals of all the other models.

We trained these three models with the same order $m_u = m_y = 1$, referring to (2). The models are trained on the training dataset following the approach described in Section III. We refer to the resulting model with the function

$$v(t+1) = f(\varphi(t)) = \sum_{i=1}^{M} \Phi_i(\varphi(t))(w_i\, \varphi(t) + \xi_i). \tag{9}$$

We test this model by comparing its output, $v(t+1)$, with the one recorded in the dataset, $v^d(t+1)$. The training results are displayed in Table I. The evaluation is based on the 1-step RMSE, obtained as:

$$\text{RMSE} = \sqrt{\frac{1}{N-1}\sum_{t=1}^{N-1} \|v(1|t) - v^d(t+1)\|_2^2}\,. \tag{10}$$

Where the notation $v(i|t)$ indicates the value of v at time $t+i$, estimated at time t.

V. Control Design

The control layout in Fig. 4 is used to control the excavator's arm assembly during the grading operation. A trajectory generator provides the reference signal in the operational space coordinates $p_r(t) = [x_r(t), z_r(t), \phi_r(t)]^\top \in \mathbb{R}^3$, where $x(t)$ and $z(t)$ represent the coordinates of the tool center point in the working plane, and $\phi(t)$ the angle of the bucket.

The *Pose Controller* - a PI controller - sets the reference velocities in operational space coordinates based on the TCP position error. Through the *Inverse Differential Kinematic* equations - known from the geometrical construction of the machine - a velocity reference for each cylinder is obtained. To track such references, we developed two distinct MPC formulations: the first uses the nonlinear LMN model, while the second formulation exploits the model structure to obtain a linear MPC at each time t, thus it can be considered as a linear time-varying (LTV) MPC law. Both formulations described below are implemented with a standard receding horizon approach [7].

A. Nonlinear MPC formulation

The finite horizon optimal control problem (FHOCP) in the nonlinear MPC formulation that we implemented is:

$$\min_{\chi} \sum_{k=0}^{N} \Big(\|v(k|t) - v_r\|_{W_v}^2 + \|j(k|t)\|_{W_j}^2 + \\ + \|j(k+1|t) - j(k|t)\|_{W_{\Delta j}}^2 \Big) \tag{11a}$$

$$\text{s.t.} \quad v(k+1|t) = f(v(k|t), j(k|t)), \qquad k \in \mathbb{N}_0^{N-1} \tag{11b}$$

$$\underline{j} \le j(k|t) \le \overline{j}, \qquad k \in \mathbb{N}_0^{N-1} \tag{11c}$$

$$v_r = v_r(t), \quad v(0|t) = v_0. \tag{11d}$$

The optimization variables are

$$\chi = [j(0|t)^\top, \dots, j(N|t)^\top, \\ v(0|t)^\top, \dots, v(N|t)^\top]^\top \in \mathbb{R}^{6(N+1)}, \tag{12}$$

and the cost function Eq. (11a) contains the reference tracking term and two terms that minimize the control effort. We use the notation $\mathbb{N}_0^{N-1}$ to indicate the set of natural numbers between 0 and $N-1$.

This problem is a standard FHOCP, in which the error between the predicted velocities $v(k|t)$, and the reference v_r is minimized. The reference is kept constant throughout the prediction horizon, since it is provided by the PI controller and future values are not available. It is thus expected to have an unavoidable lag in the reference tracking.

The input signals are constrained between their lower bound $\underline{j}$ and upper bound $\overline{j}$ in Eq. (11c), which are chosen to match the boundaries of the values in the training dataset, avoiding model extrapolation.

B. LTV-MPC implementation

For the LTV-MPC, we use the same FHOCP problem formulation as in Section V-A, but with a linearized model function around the current state and input $\hat{\varphi}$: $f\big|_{\hat{\varphi}}(\varphi)$. An approximate linearization of the model around the current state and input $\hat{\varphi} = [\hat{j}^\top, \hat{v}^\top]^\top$ can be obtained exploiting the structure of the LMN. We start by recalling that the output of the network at a specific point is the weighted average of the output of the individual LLMs. From this, if we fix the weight of each local model, i.e. its membership function Φ_i, we can obtain a linear model, whose coefficients are the

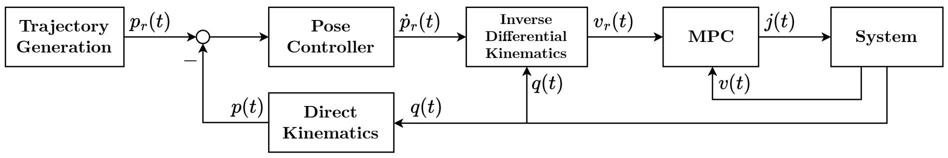

Fig. 4: Proposed control scheme, where the MPC law employs either the derived Local Model Network (`nMPC` scheme) or its linear approximation updated in real time (`lMPC` scheme). Here, vector q indicates the hydraulic cylinders' displacements, p the TCP position in the operational space, v the cylinders' velocities, and j the joystick signals.

weighted average of the ones of the LLMs:

$$f|_{\hat{\varphi}}(\varphi) = \sum_{i=1}^{M} \Phi_i(\hat{\varphi})(w_i\varphi(t) + \xi_i) = \sum_{i=1}^{M} \Phi_i(\hat{\varphi})w_i\varphi(t) + \sum_{i=1}^{M} \Phi_i(\hat{\varphi})\xi_i = w|_{\hat{\varphi}}\,\varphi(t) + \xi|_{\hat{\varphi}}.$$

The linearization is not exact, since it neglects the derivatives of the membership functions, however, this approximation proved to have a negligible impact on the quality of the prediction. The resulting FHOCP, using this model, is:

$$\min_{\chi} \sum_{k=0}^{N} \Big(\|v(k|t) - v_{\mathrm{r}}\|^2_{W_v} + \|j(k|t)\|^2_{W_j} + \|j(k+1|t) - j(k|t)\|^2_{W_{\Delta j}} \Big) \tag{13a}$$

s.t.

$$v(k+1|t) = f|_{v(t),j(t)}(v(k|t), j(k|t)), \quad k \in \mathbb{N}_0^{N-1} \tag{13b}$$

$$\underline{j} \le j(k|t) \le \overline{j}, \quad k \in \mathbb{N}_0^{N-1} \tag{13c}$$

$$v_{\mathrm{r}} = v_{\mathrm{r}}(t), \quad v(0|t) = v_0, \tag{13d}$$

which reflects what we presented in Eq. (11). However, thanks to the linear dynamics function, this FHOCP is now a linearly-constrained quadratic problem, therefore simpler to solve. This leads to faster online computation, aspect of paramount importance when implementing the controller on embedded hardware.

VI. Experimental Results

We tested the proposed MPC controllers on the experimental platform depicted in Fig. 1, in grading operations, as shown in Fig. 2. The test excavator is equipped with a *dSpace MicroAutoBox II* embedded prototyping device, that allows to quickly and efficiently deploy and test different controllers. We compared the MPC approaches (`nMPC` and `lMPC`) with a state of the art, neural network (NN) -based reference controller (`ref`). Each controller was executed with a different rate, 100ms for `nMPC`, 40ms for `lMPC`, and 10ms for the reference controller.

A. Benchmark Controller

The `ref` controller was previously developed in [4]. It uses a similar control scheme to the one depicted in Fig. 4, the difference lies in the velocity control function. This consists of three distinct controllers, one for each hydraulic axis. For the bucket a simple model-based controller is used. For the boom and the arm hydraulic cylinders, two NNs are trained to capture the inverse system behaviour, and then used as feed-forward controllers.

Param.	Value	**Parameter**	Value
		`nMPC`	
N	5	NLP Solver	SQP
W_v	10	Max. Iter. SQP	7
W_j	3	QP Solver	`HPIPM`
$W_{\Delta j}$	1	Max. Iter. QP	50
		`lMPC`	
N	10	NLP Solver	-
W_v	10	Max. Iter. SQP	-
W_j	3	QP Solver	`HPIPM`
$W_{\Delta j}$	1	Max. Iter. QP	50

TABLE II: `nMPC` and `lMPC` controller parameters used for the presented experimental results.

To facilitate a comparison between the reference controller and the proposed approach, three distinct variants were developed. In the first variant, the NNs are trained on the same dataset used for the LNMs, i.e. `Dataset-A`. For the second variant, the NNs are trained on `Dataset-B`, which expands upon `Dataset-A` by including additional measurements of Δp, the pressure difference between the pump and the Load Sensing system. The final variant uses `Dataset-C`, a substantially different dataset containing approximately 150 minutes of data collected through a similar procedure as the one described in Section IV-A. This last dataset excludes any leveling data while containing Δp measurements.

B. MPC Implementation and results

We implemented the MPC controllers using the `acados` toolbox, presented in [13], and its Matlab interface. This allows to establish a pipeline that, starting from the datasets, trains the LMNs using the LoLiMoT algorithm, then generates `c` code functions of the FHOCP, and finally compiles them for the dSpace platform. This workflow requires minimal input from the user, and results in a computationally efficient implementation that can be executed in real-time on the experimental platform.
For the experimental results, the chosen horizon length N and the cost function weights $W_v, W_j, W_{\Delta j}$, together with the solver's parameters, are depicted in Table II.

The results of the tests are collected in Table III, where the nonlinear MPC (`nMPC`) and the linear time-varying MPC (`lMPC`) are compared with the three variants of the reference controller (`ref`). All tests have been conducted with the same position reference $p_{\mathrm{r}}(t)$. On the other hand, the

RMSE [m/s]	Dataset	Boom	Arm	Bucket
nMPC	Dataset-A	0.0036	0.0088	0.0043
lMPC	Dataset-A	0.0042	0.0087	0.0046
ref	Dataset-A	0.0052	0.0093	0.0127
ref	Dataset-B	0.0046	0.0074	0.0062
ref	Dataset-C	0.0042	0.0087	0.0053

TABLE III: Velocity tracking results for the three controllers.

reference velocity $\dot{p}_{\mathrm{r}}(t)$ (see Fig. 4) is generally not identical in all tests, because it is generated by the pose controller and varies based on the closed-loop behavior of the system.

An analysis of the three reference controller variants shows that the dataset used to train the NNs has a considerable impact on performance. When trained on Dataset-A, the tracking accuracy of ref is significantly worse than that of both nMPC and lMPC. Accuracy improves with Dataset-B, highlighting the importance of Δp for this approach. Further improvement is seen with the larger Dataset-C, where the accuracy becomes comparable to that of nMPC and lMPC, still using the differential pressure as further feedback variable.

Shifting the attention on lMPC, we notice a very slight decrease in velocity tracking accuracy with respect to nMPC, mainly on the boom axis, which is the one that experiences the strongest non-linearities. This is in line with the expectations of using an approximation of the system model. Interestingly, the effect of the loss in prediction accuracy is almost fully compensated by the lower sampling rate, that allows for a faster closed-loop reaction to errors.
Regarding the computation times, we showed how the proposed pipeline is able to produce a controller implementation with real-time capabilities on embedded hardware. The measured average computation times for nMPC and lMPC were 94ms and 38ms, respectively, both of which are below their corresponding execution intervals of 100ms and 40ms.
We can also notice how the linear approximation of the system model, and the subsequent implementation of a linearly constrained quadratic FHOCP, allows to reduce by almost 60% the computation effort required.
This aspect is relevant for an implementation in series product, where the controllers need to be trained and adapted for each individual product, to compensate for the high variability.

VII. Conclusions

Novel approaches to design an assistive grading function for excavators have been proposed. An accurate nonlinear model of the system, which captures the relationship between the joystick signals and the velocities of the hydraulic cylinders, was derived from data using the LoLiMoT algorithm. This nonlinear model was then used as prediction model for a nonlinear velocity tracking MPC. The results on the real-time system demonstrated that the proposed controller met the same performance as the benchmark controller, while using a smaller dataset during training and without using any information relative to the pump pressure.

Taking advantage of the underlying linear models of the trained LMN, a linear time-varying formulation of the MPC was developed. The experiments on the real system showed that this controller is much faster than the nonlinear counterpart, while the accuracy of the tracking did not decrease. This proposed controller is thus computationally less demanding, making it particularly appealing for production units, as it eliminates the requirement for a powerful embedded device.

Finally, the proposed approach also eliminated the need to use Δp. This is particularly relevant for the industry as it potentially removes the requirement for an additional sensor, reducing hardware costs and system complexity.

Future work may include the use of multi-step prediction error in the model training, to take into account the error accumulation, and the development of an Adaptive MPC.

References

[1] P. Egli and M. Hutter, "Towards RL-Based Hydraulic Excavator Automation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 2692–2697.

[2] J. Nurmi, M. M. Aref, and J. Mattila, "A Neural Network Strategy for Learning of Nonlinearities Toward Feed-Forward Control of Pressure-Compensated Hydraulic Valves With a Significant Dead Zone," in *BATH/ASME 2018 Symposium on Fluid Power and Motion Control*. Bath, UK: American Society of Mechanical Engineers, Sep. 2018, p. V001T01A023.

[3] T. Kinoshita, K. Koiwai, T. Yamamoto, T. Nanjo, Y. Yamazaki, and Y. Fujimoto, "Design of a Data-Driven Control System for a Hydraulic Excavator," *Proceedings of International Conference on Artificial Life and Robotics*, vol. 21, pp. 393–396, Jan. 2016.

[4] J. Weigand, J. Raible, N. Zantopp, O. Demir, A. Trachte, A. Wagner, and M. Ruskowski, "Hybrid Data-Driven Modelling for Inverse Control of Hydraulic Excavators," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021, pp. 2127–2134.

[5] G. Rabenstein, O. Demir, A. Trachte, and K. Graichen, "Data-Driven Feed-Forward Control of Hydraulic Cylinders using Gaussian Process Regression for Excavator Assistance Functions," in *2022 IEEE Conference on Control Technology and Applications (CCTA)*. Trieste, Italy: IEEE, Aug. 2022, pp. 962–969.

[6] M. Dio, O. Demir, A. Trachte, and K. Graichen, "Safe Active Learning and Probabilistic Design of Experiment for Autonomous Hydraulic Excavators," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Detroit, MI, USA: IEEE, Oct. 2023, pp. 9685–9690.

[7] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[8] F. A. Bender, M. Sonntag, and O. Sawodny, "Nonlinear model predictive control of a hydraulic excavator using Hammerstein models," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. Queenstown, New Zealand: IEEE, Feb. 2015, pp. 557–562.

[9] F. A. Bender, S. Goltz, T. Braunl, and O. Sawodny, "Modeling and Offset-Free Model Predictive Control of a Hydraulic Mini Excavator," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1682–1694, Oct. 2017.

[10] O. Nelles, *Nonlinear System Identification*, 2nd ed. Springer, 2020.

[11] O. Nelles, A. Fink, and R. Isermann, "Local linear model trees (lolimot) toolbox for nonlinear system identification," *IFAC Proceedings Volumes*, vol. 33, no. 15, pp. 845–850, 2000.

[12] O. Nelles and R. Isermann, "A comparison between rbf networks and classical methods for identification of nonlinear dynamic systems," *IFAC Proceedings Volumes*, vol. 28, no. 13, pp. 233–238, 1995, 5th IFAC Symposium on Adaptive Systems in Control and Signal Processing 1995, Budapest, Hungary, 14-16 June, 1995.

[13] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, pp. 147–183, Oct 2021.