Structurally Optimized Neural Fuzzy Modelling for Model Predictive Control

Xiaoyan Hu, Yu Gong, Dezong Zhao, *Senior Member, IEEE,* and Wen Gu

***Abstract*—This paper investigates the local linear model tree (LOLIMOT), a typical neural fuzzy model, in the multiple-input-multiple-output model predictive control (MPC). In the conventional LOLIMOT, the structural parameters including centres and variances of its Gaussian kernels are set based on equally dividing the input data space. In this paper, after the structural parameters are initially obtained from the input space partition, they are optimized by the gradient descent search, from which the space partitions are further adjusted. This makes it better for the model structure to fit the input data statistics, leading to improved modelling performance with small model size. The MPC based on the proposed structurally optimized LOLIMOT is then implemented and verified with both numerical and diesel engine plants. Validation results show that the proposed MPC has significantly better controlling performance than the MPC based on the conventional LOLIMOT, making it an attractive solution in practice.**

***Index Terms*—Neural fuzzy network, local linear model tree, multiple-input-multiple-output nonlinear system, model predictive control.**

I. Introduction

MODEL predictive control (MPC) has been used in many fields including robotics, vehicle, aerospace and power electronics [1]. The performance of the MPC well depends on modelling the target plant because of the model-based optimization to obtain the control actions [2]. Both linear MPC [3] and nonlinear MPC (NMPC) [4] have been proposed.

The neural network model have been well investigated in the NMPC [5]. Of particular interest is the neural fuzzy network which consists of a number of linear subsystems with membership functions, where the membership functions describe fuzzy rules to combine the linear subsystems [6]. The neural fuzzy models have been widely used in the control system. Examples include the MPC with finite horizon based on the fuzzy discrete systems [7], the output feedback predictive control based on the Takagi–Sugeno (T–S) fuzzy model [8] [9], the NMPC based on self-feedback fuzzy network [10], and the adaptive T–S fuzzy model-based predictive controller [11]. The neural fuzzy network is also used in the intelligent control [12], [13].

A number of neural fuzzy models have been proposed including the interval type-2 radial basis function neural network [14], T-S fuzzy system [15], data driven modelling based on fuzzy neural network [16] and recurrent fuzzy neural network [17]. This paper investigates the local linear model tree (LOLIMOT) [18] [19], a typical neural fuzzy model. The LOLIMOT consists of a number of neurons, where each neuron includes a Gaussian kernel and a local linear structure. Every Gaussian kernel defines a membership function representing a fuzzy rule. The performance of the LOLIMOT relies on the model structure including the model size (i.e. the neuron number), the centers and variances of the Gaussian kernels. In the LOLIMOT, the incremental tree structure is used to divide the input data space into grid partitions. Every partition determines the structural parameters of one neuron that the centers and variances of the corresponding Gaussian kernel are the centers and proportional to the widths of the partition, respectively. The LOLIMOT starts with a single neuron, and repeatedly generates new neurons by dividing an existing partition equally into two until the satisfactory modelling performance is reached. While equally dividing existing partitions provides a simple way to generate new neurons in the LOLIMOT, the resulted neuron structure may not match the input data statistics. Non-appropriately structured neurons often lead to large model size, resulting in over-fitting and high complexity. This is because large neuron number complicates the optimization in the MPC.

The structure of the neural fuzzy model can be optimized with the particle swarm optimization (PSO) [20] and genetic algorithm (GA) [21]. These approaches however demand high complexity. In this paper, we propose a simple yet efficient way based on the gradient descent search to optimize the LOLIMOT structure. The proposed structurally optimized LOLIMOT achieves good performance with small model size, making it particularly attractive for the NMPC. The main contribution of this paper is listed as following:

- A novel gradient descent search approach is proposed to optimize centers and variances of the Gaussian kernels, based on which the input space partitions are further adjusted. This leads to more efficient Gaussian kernels in the LOLIMOT.
- The NMPC based on the proposed structurally optimized LOLIMOT model is implemented.
- The proposed NMPC is verified by both numerical data and experimental data from the turbocharged diesel engine platform, where significantly better control perfor-

This work was supported by the Engineering and Physical Sciences Research Council of the U.K., through the EPSRC Innovation Fellowship, under Grant EP/S001956/2.

X. Hu, Y. Gong are with the Wolfson School, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: x.hu4@lboro.ac.uk, y.gong@lboro.ac.uk).

D. Zhao is with James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K. (e-mail:dezong.zhao@glasgow.ac.uk).

W. Gu is with the Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough LE11 3TU, U.K. (e-mail: w.gu@lboro.ac.uk).

mance is observed in both cases.

II. Neural Fuzzy Network

A. System model

The multiple-input-multiple-output (MIMO) neural fuzzy model consisting of M neurons as shown in Fig. 1, where

$$\hat{y}_i(k+1) = \sum_{m=1}^{M} \hat{y}_i^m(k+1), \qquad i = 1, \cdots, q \tag{1}$$

is the i-th model output, $\hat{y}_i^m(k+1)$ is the i-th output of the m-th neuron, and $\boldsymbol{\xi}(k)$ is the model input vector:

$$\boldsymbol{\xi}(k) = [\hat{\boldsymbol{y}}_1^\top(k), ..., \hat{\boldsymbol{y}}_q^\top(k), \boldsymbol{u}_1^\top(k), ..., \boldsymbol{u}_p^\top(k)]^\top \tag{2}$$

with

$$\begin{aligned} \hat{\boldsymbol{y}}_i(k) &= [\hat{y}_i(k), \hat{y}_i(k-1), ..., \hat{y}_i(k-n_{y,i}+1)]^\top \\ \boldsymbol{u}_j(k) &= [u_j(k), u_j(k-1), ..., u_j(k-n_{u,j}+1)]^\top \end{aligned} \tag{3}$$

where $i = 1, 2, ..., q$, $j = 1, 2, ..., p$, $n_{y,i}$ and $n_{u,j}$ are the time lags for the feedback output and input signals, respectively. The input dimension of the model is the number of elements in $\boldsymbol{\xi}(k)$ which is given by

$$L = \sum_{i=1}^{q} n_{y,i} + \sum_{j=1}^{p} n_{u,j} \tag{4}$$

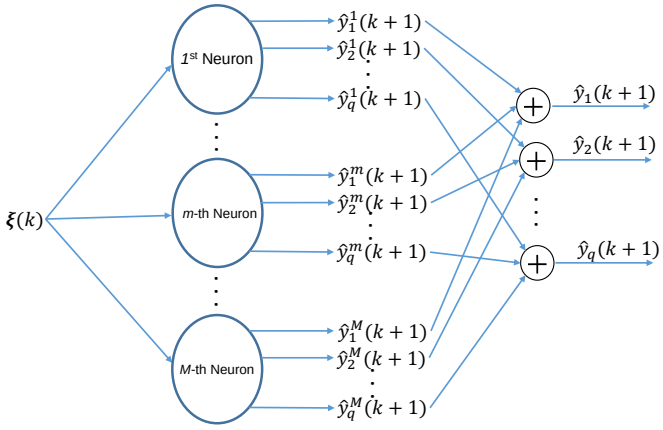


Fig. 1: The MIMO neural fuzzy network

The m-th neuron is shown in Fig. 2 which consists of one Gaussian kernel and q local linear models (LLM). The input vector of the m-th Gaussian kernel is $\boldsymbol{\xi}(k)$, and the output is

$$\phi_m(k) = \frac{\mu_m(k)}{\sum_{j=1}^{M} \mu_j(k)} \tag{5}$$

where

$$\mu_m(k) = \exp\left(-\frac{1}{2} \cdot \sum_{l=1}^{L} \left(\frac{(\xi_l(k) - c_{m,l})^2}{\sigma_{m,l}^2}\right)\right) \tag{6}$$

$c_{m,l}$ and $\sigma_{m,l}$ are the center and standard deviation of the Gaussian kernel in the l-th dimension respectively, and $\xi_l(k)$ is the l-th element of $\boldsymbol{\xi}(k)$. On the other hand, the input of the LLM is the augmented input vector which is given by

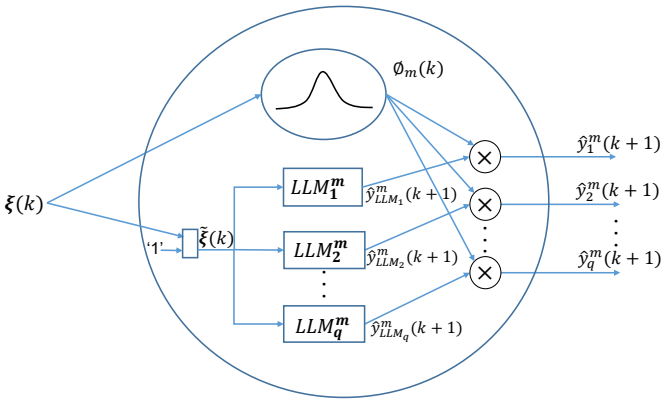


Fig. 2: The m-th neuron of the MIMO neural fuzzy network

$$\widetilde{\boldsymbol{\xi}}(k) = [1, \boldsymbol{\xi}^\top(k)]^\top, \tag{7}$$

and the output of the i-th LLM is given by

$$\hat{y}_{\mathrm{LLM}_i}^m(k+1) = \boldsymbol{\omega}_{m,i}^\top \cdot \widetilde{\boldsymbol{\xi}}(k) \tag{8}$$

where $\boldsymbol{\omega}_{m,i}$ is weight vector for the i-th LLM:

$$\boldsymbol{\omega}_{m,i} = [\omega_{m,i,0}, a_{m,i,1}, a_{m,i,2}, ..., b_{m,i,1}, b_{m,i,2}, ...]^\top \tag{9}$$

where coefficients a and b correspond to the corresponding feedback output y_i and data input u_j, respectively, and $\omega_{m,i,0}$ is the offset.

The Gaussian kernel determines the contribution of the corresponding LLM, and so the i-th output of the m-th neuron is obtained by

$$\hat{y}_i^m(k+1) = \phi_m(k) \cdot \hat{y}_{\mathrm{LLM}_i}^m(k+1) \tag{10}$$

As shown in Fig. 1, summing the i-th output from all neurons gives the i-th output of the overall neural-fuzzy system:

$$\hat{y}_i(k+1) = \sum_{m=1}^{M} \phi_m(k) \cdot \hat{y}_{\mathrm{LLM}_i}^m(k+1) \tag{11}$$

B. Local linear model tree

The LOLIMOT describes an efficient way to realize the neural fuzzy model, in which the centres and variances of the Gaussian kernels are set based on dividing the input space of the training data. Assume there are K snapshots of training data set denoted as $\{\boldsymbol{u}(k),\ \boldsymbol{y}(k+1)\}$, $k = 1, \cdots, K$, where $\boldsymbol{u}(k) = [u_1(k), \cdots, u_p(k)]^\top$ and $\boldsymbol{y}(k+1) = [y_1(k+1), \cdots, y_q(k+1)]^\top$. In the LOLIMOT with M neurons, the input data space is divided into M hyper rectangular partitions based on the training data set. Every partition corresponds to one neuron, where the center and variance of a partition decide the center and variance of the corresponding Gaussian kernel. The training starts with a single partition (which corresponds to a single neuron), and more partitions are created iteratively until the global loss function satisfies the requirement or the maximum number of neurons is reached.

The global loss function is defined as

$$G = \sum_{i=1}^{q} G_i \tag{12}$$

where G_i is loss function for the q-th output:

$$G_i = \gamma_i \cdot \sum_{k=1}^{K} [y_i(k+1) - \hat{y}_i(k+1)]^2, \tag{13}$$

$\gamma_1, ..., \gamma_q$ are the weight coefficients that $\gamma_1 + \gamma_2 + ... + \gamma_q = 1$. Suppose there are m neurons (or m partitions) at the $(i-1)$-th iteration. At the i-th iteration, if G is higher than the pre-defined threshold and the maximum number of neurons is not reached, the current partition with the highest ϕ_m defined in (5) is selected to be equally divided into two partitions. Fig. 3 illustrates how the selected partition is divided along the j-th input dimension equally into two new partitions, m_1 and m_2, corresponding to two new neurons m_1 and m_2, respectively. As shown in Fig. 3, $\boldsymbol{A}_j$ and $\boldsymbol{B}_j$ are vertex coordinate vectors that determine the selected partition, and $\boldsymbol{A}'_j$ and $\boldsymbol{B}'_j$ are new vertex coordinate vectors of the new partitions.

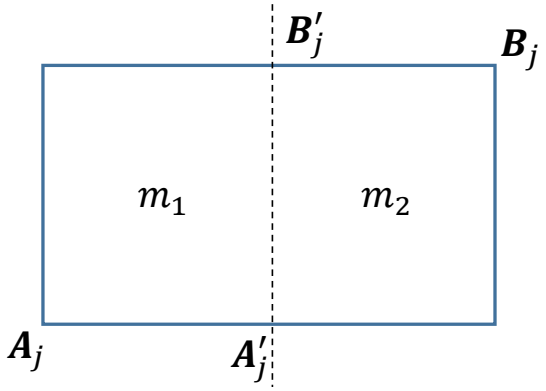


Fig. 3: Split on the j-th dimension for the neuron

The center and standard deviation vectors of the Gaussian kernels for the two new neurons m_1 and m_2 are obtained as

$$\boldsymbol{c}_{m_1} = (\boldsymbol{A}_j + \boldsymbol{B}'_j) \cdot 0.5 \tag{14a}$$
$$\boldsymbol{c}_{m_2} = (\boldsymbol{A}'_j + \boldsymbol{B}_j) \cdot 0.5 \tag{14b}$$
$$\boldsymbol{\sigma}_{m_1} = (\boldsymbol{B}'_j - \boldsymbol{A}_j) \cdot \eta \tag{14c}$$
$$\boldsymbol{\sigma}_{m_2} = (\boldsymbol{B}_j - \boldsymbol{A}'_j) \cdot \eta \tag{14d}$$

respectively, where η is the smoothness factor for the Gaussian variance. The coefficient vector of the i-th LLM in (9) for the new neuron is obtained with the weighted least squares (WLS):

$$\boldsymbol{\omega}_{m_s,i} = (\boldsymbol{\zeta}^\top \boldsymbol{Q}_{m_s} \boldsymbol{\zeta})^{-1} \boldsymbol{\zeta}^\top \boldsymbol{Q}_{m_s} \boldsymbol{y}_i, m_s \in \{m_1, m_2\} \tag{15}$$

where $\boldsymbol{\zeta} = [\widetilde{\boldsymbol{\xi}}^\top(1), \cdots, \widetilde{\boldsymbol{\xi}}^\top(K)]^\top$ is the data matrix, $\boldsymbol{Q}_{m_s}$ = diag$[\phi_{m_s}(\boldsymbol{\xi}(1)), \cdots, \phi_{m_s}(\boldsymbol{\xi}(K))]$ is the weight matrix and $\boldsymbol{y}_i = [y_i(2), \cdots, y_i(K+1)]^\top$ is the training output data vector for the i-th output.

From (14) and (15), the global loss function G with the newly generated neurons m_1 and m_2 can be obtained. Applying the above procedure to every input dimension, the partition along the input dimension with the lowest G is then used to generate the new neurons.

III. Structurally optimized LOLIMOT

In the LOLIMOT, the centers and variances of the Gaussian kernels are chosen as the centers and widths (with the smooth factor) of the corresponding input space partitions. However, this may not reflect the underlying statistics of the input data, leading to modelling performance degradation. In this section, the centers and variances of the Gaussian kernels are further optimized using the gradient descent search.

Like in the LOLIMOT, the centers and variances of the Gaussian kernels are also obtained based on the input space partition. Similarly as above, suppose there are m neurons (corresponding to m partitions) at the $(i-1)$-th iteration. As shown in Fig. 3, at the i-th iteration, the selected partition (with the highest ϕ_m) is equally divided into two partitions, m_1 and m_2, along the j-th input dimension, respectively. The centers and variances of the Gaussian kernels for the two new partitions are obtained as in (14), which will be used as the initial values for the gradient descent search.

The center and standard derivation vectors of the Gaussian kernels corresponding to the new partition m_1 and m_2 are adapted using the gradient descent search as

$$\boldsymbol{c}_{m_s}(t+1) = \boldsymbol{c}_{m_s}(t) - \beta \cdot \nabla_{\boldsymbol{c}_{m_s}(t)} G(t) \tag{16}$$

$$\boldsymbol{\sigma}_{m_s}(t+1) = \boldsymbol{\sigma}_{m_s}(t) - \beta \cdot \nabla_{\boldsymbol{\sigma}_{m_s}(t)} G(t) \tag{17}$$

respectively, where the index t represents the t-th adaptive iteration, β is the step size, ∇ is the gradient operator, and $m_s \in \{m_1, m_2\}$ correspond to the new partitions m_1 and m_2, respectively.

From (12) we have

$$\nabla_{\boldsymbol{c}_{m_s}(t)} G(t) = \frac{\partial G(t)}{\partial \boldsymbol{c}_{m_s}(t)} = \sum_{i=1}^{q} \frac{\partial G_i(t)}{\partial \boldsymbol{c}_{m_s}(t)} \tag{18}$$

$$\nabla_{\boldsymbol{\sigma}_{m_s}(t)} G(t) = \frac{\partial G(t)}{\partial \boldsymbol{\sigma}_{m_s}(t)} = \sum_{i=1}^{q} \frac{\partial G_i(t)}{\partial \boldsymbol{\sigma}_{m_s}(t)} \tag{19}$$

where from (13) we have

$$\frac{\partial G_i(t)}{\partial \boldsymbol{c}_{m_s}(t)} = -2 \cdot \gamma_i \cdot \sum_{k=1}^{K} [y_i(k+1) - \hat{y}_i(k+1)] \cdot \frac{\partial \hat{y}_i(k+1)}{\partial \boldsymbol{c}_{m_s}(t)} \tag{20}$$

$$\frac{\partial G_i(t)}{\partial \boldsymbol{\sigma}_{m_s}(t)} = -2 \cdot \gamma_i \cdot \sum_{k=1}^{K} [y_i(k+1) - \hat{y}_i(k+1)] \cdot \frac{\partial \hat{y}_i(k+1)}{\partial \boldsymbol{\sigma}_{m_s}(t)} \tag{21}$$

Further from (5), (6) and (10), we have

$$\begin{aligned} \frac{\partial \hat{y}_i(k+1)}{\partial \boldsymbol{c}_{m_s}(t)} &= \hat{y}^{m_s}_{\text{LLM}_i}(k+1,t) \cdot \\ &\frac{\frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{c}_{m_s}(t)} \cdot \sum_{m=1}^{M} \mu_m(k,t) - \mu_{m_s}(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{c}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} \\ &+ \hat{y}^{\overline{m}_s}_{\text{LLM}_i}(k+1,t) \cdot \frac{-\mu_{\overline{m}_s}(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{c}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} \\ &+ \sum_{\substack{m=1, \\ m \notin \{m_1, m_2\}}}^{M} \hat{y}^{m}_{\text{LLM}_i}(k+1,t) \cdot \frac{-\mu_m(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{c}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} \end{aligned} \tag{22}$$

and

$$\frac{\partial \hat{y}_i(k+1)}{\partial \boldsymbol{\sigma}_{m_s}(t)} = \hat{y}^{m_s}_{\text{LLM}_i}(k+1,t)\cdot \frac{\frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{\sigma}_{m1}(t)} \cdot \sum_{m=1}^{M} \mu_m(k,t) - \mu_{m_s}(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{\sigma}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} + \hat{y}^{\overline{m}_s}_{\text{LLM}_i}(k+1,t) \cdot \frac{-\mu_{\overline{m}_s}(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{\sigma}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} + \sum_{\substack{m=1,\\ m\notin\{m_1,m_2\}}}^{M} \hat{y}^{m}_{\text{LLM}_i}(k+1,t) \cdot \frac{-\mu_m(k,t) \cdot \frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{\sigma}_{m_s}(t)}}{[\sum_{m=1}^{M} \mu_m(k,t)]^2} \tag{23}$$

where $\overline{m}_s$ is the complement of m_s that $\overline{m}_s = m_1$ if $m_s = m_2$, and $\overline{m}_s = m_2$ if $m_s = m_1$. From (6) we have

$$\frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{c}_{m_s}(t)} = \left[\frac{\partial \mu_{m_s}(k,t)}{\partial c_{m_s,1}(t)}, \cdots, \frac{\partial \mu_{m_s}(k,t)}{\partial c_{m_s,L}(t)}\right]^\top = \left[\mu_{m_s}(k,t) \cdot \frac{\xi_1(k) - c_{m_s,1}(t)}{\sigma^2_{m_s,1}(t)}, \cdots, \mu_{m_s}(k,t) \cdot \frac{\xi_L(k) - c_{m_s,L}(t)}{\sigma^2_{m_s,L}(t)}\right]^\top \tag{24}$$

and

$$\frac{\partial \mu_{m_s}(k,t)}{\partial \boldsymbol{\sigma}_{m_s}(t)} = \left[\frac{\partial \mu_{m_s}(k,t)}{\partial \sigma_{m_s,1}(t)}, \cdots, \frac{\partial \mu_{m_s}(k,t)}{\partial \sigma_{m_s,L}(t)}\right]^\top = \left[\mu_{m_s}(k,t) \cdot \frac{[\xi_1(k) - c_{m_s,1}(t)]^2}{\sigma^3_{m_s,1}(t)}, \cdots, \mu_{m_s}(k,t) \cdot \frac{[\xi_L(k) - c_{m_s,L}(t)]^2}{\sigma^3_{m_s,L}(t)}\right]^\top \tag{25}$$

The LLM weight vectors for the two new partitions are updated by the WLS as in (15). Apply the Gaussian parameters adaptation on every input dimension, and choose the dimension with the lowest global loss function G to divide the selected partition for the new neurons. After the centers and variances are adapted, the partition boundary separating the two new partitions is adjusted. The new partition boundary is the middle of the adapted centers of the two new partitions. Continue generating new partitions as above until the global loss function G is smaller than the pre-defined value G' or the maximum number of the neurons (M_{max}) is reached. The proposed algorithm is summarized in Algorithm 1.

Algorithm 1: The algorithm to optimize the Gaussian parameters in the LOLIMOT

```
Initialization;
Start with the first neuron;
while M < M_max or G > G' do
    Select the neuron with the highest φ_m as in (5);
    for every input dimension (j = 1 : L) do
        Obtain the initial centers and standard deviations for the new kernels m_1 and m_2 as in (14)
        for every gradient descent search iteration (i = 1 : Z) do
            Update the c_{m_1}, c_{m_2}, σ_{m_1}, σ_{m_2} using (16) and (17);
            Update the LLM-s vectors ω_{m_1,i} and ω_{m_2,i} as in (15) based on the updated centers and standard deviations;
        end
        Obtain the global loss function G for the j-th dimension as in (12);
    end
    Choose the dimension with the smallest G to divide the selected partition;
    Adjust the partition boundary separating the two new partitions as the middle of the two adapted centers.
end
```

IV. Neural Fuzzy Model Predictive Control

In this section, after the proposed structurally optimized LOLIMOT is trained as above, it is applied in the MIMO predictive control. The block diagram of the so-called MIMO neural fuzzy based predictive control (MIMO-NFMPC) is shown in Fig. 4.

Assuming the prediction horizon is N_p, at time k, the estimated plant input, the corresponding plant output and the set point vectors are given by

$$\boldsymbol{u}^{(N_p)}(k) = [\boldsymbol{u}^\top(k), \cdots, \boldsymbol{u}^\top(k+N_p-1)]^\top \tag{26}$$

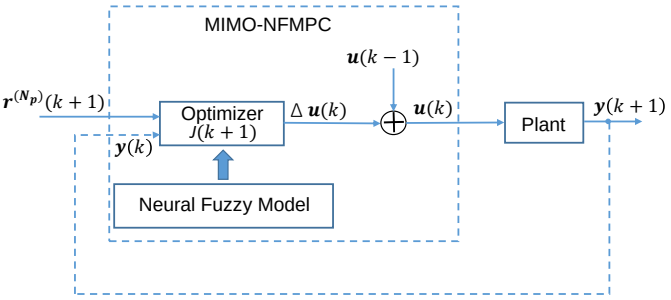


Fig. 4: The block diagram of the MIMO-NFMPC

$$\boldsymbol{y}^{(N_p)}(k+1) = [\hat{\boldsymbol{y}}^\top(k+1), \cdots, \hat{\boldsymbol{y}}^\top(k+N_p)]^\top \tag{27}$$

$$\boldsymbol{r}^{(N_p)}(k+1) = [\boldsymbol{r}^\top(k+1), \cdots, \boldsymbol{r}^\top(k+N_p)]^\top \tag{28}$$

where $\boldsymbol{u}(k) = [u_1(k), \cdots, u_p(k)]^\top$, $\hat{\boldsymbol{y}}(k) = [\hat{y}_1(k), \cdots, \hat{y}_q(k)]^\top$ and $\boldsymbol{r}(k) = [r_1(k), \cdots, r_q(k)]^\top$, while p and q are the input and output dimension of the plant.

At time k, the plant setpoint for the next N_p steps, $\boldsymbol{r}^{(N_p)}(k+1)$, is set as in (28). By observing the current plant output $\mathbf{y}(k)$, the MPC estimates the plant inputs for the next N_p steps, $\boldsymbol{u}^{(N_p)}(k)$, such that the predicted plant output based on the trained LOLIMOT model in the future N_p steps, $\boldsymbol{y}^{(N_p)}(k+1)$, can best track the setpoints $\boldsymbol{r}^{(N_p)}(k+1)$. This is achieved by minimizing the cost function

$$J(k+1) = ||\boldsymbol{r}^{(N_p)}(k+1) - \boldsymbol{y}^{(N_p)}(k+1)||^2 + \varrho \cdot ||\Delta \boldsymbol{u}^{(N_p)}(k)||^2 \tag{29}$$

subject to

$$\begin{aligned} u_{i,\min} \le u_i(j) \le u_{i,\max}, \\ \Delta u_{i,\min} \le \Delta u_i(j) \le \Delta u_{i,\max}, \qquad \forall\ i,j \\ y_{m,\min} \le y_m(K+n) \le y_{m,\max}, \quad m \in \{1,q\}, n \in \{1,N_p\} \end{aligned} \tag{30}$$

where

$$\Delta \boldsymbol{u}^{(N_p)}(k) = \boldsymbol{u}^{(N_p)}(k) - \boldsymbol{u}^{(N_p)}(k-1) \tag{31}$$

where $||.||$ is the 2-norm operator, ϱ is the regularization factor, and the subscripts 'min' and 'max' denote the lower and upper bounds for the corresponding parameters.

In order to achieve the offset-free reference tracking, the optimization in (29) is with respect to the incremental input $\Delta \boldsymbol{u}^{(N_p)}(k)$ defined in (31), and the plant input can be easily obtained as shown in Fig. 4. In the MPC, while the optimization is for the future N_p step, only the plant input at the current time k is applied to control the plant.

The optimization of (29) is based on the proposed structurally optimized LOLIMOT model. Because the optimization is for $\Delta \boldsymbol{u}^{(N_p)}(k)$, the model output (i.e. the estimated plant output) vector for the future N_p steps $\boldsymbol{y}^{(N_p)}(k+1)$ in the cost function $J(k+1)$ is expressed as (see Appendix A)

$$\begin{aligned} \boldsymbol{y}^{(N_p)}(k+1) = \boldsymbol{\Gamma}_1(k) \cdot \widetilde{\boldsymbol{y}}(k) + \boldsymbol{\Gamma}_2(k) \cdot \Delta \boldsymbol{u}^{(N_p)}(k) + \\ \boldsymbol{\Gamma}_3(k) \cdot \Delta \widetilde{\boldsymbol{u}}(k-1) \end{aligned} \tag{32}$$

where $\bar{\boldsymbol{\Gamma}}_{\mathbf{1}}(k)$, $\bar{\boldsymbol{\Gamma}}_{\mathbf{2}}(k)$ and $\bar{\boldsymbol{\Gamma}}_{\mathbf{3}}(k)$ are matrices independent of $\Delta \boldsymbol{u}^{(N_p)}(k)$, $\widetilde{\boldsymbol{y}}(k)$ and $\Delta \widetilde{\boldsymbol{u}}(k-1)$ are the previous model output and incremental plant input which are given by

$$\begin{aligned} \widetilde{\boldsymbol{y}}(k) = [\hat{\boldsymbol{y}}^\top(k), \cdots, \hat{\boldsymbol{y}}^\top(k - n_y^{\max})]^\top \\ \Delta \widetilde{\boldsymbol{u}}(k-1) = [\Delta \boldsymbol{u}^\top(k-1), \cdots, \Delta \boldsymbol{u}^\top(k - n_u^{\max} + 1)]^\top \end{aligned} \tag{33}$$

respectively, where $n_y^{\max} = \max\{n_{y,i}, i = 1, \cdots, q\}$, $n_u^{\max} = \max\{n_{u,j}, j = 1, \cdots, p\}$, $n_{y,i}$ and $n_{u,j}$ are defined in (3). Substituting (32) into (29) results in the constraint least square optimization which can be easily solved by existing toolboxes such as the YALMIP [22]. On the other hand, for the off-line LOLIMOT, the computational complexity is $O(2ML^4)$, where M is the number of neurons and L is the input data dimension.

V. Simulations

This section verifies the proposed MIMO-NFMPC based on the structurally optimized LOLIMOT model. Two cases are investigated, numerical plant and turbocharged diesel engine platform, respectively.

The performance of modelling is measured by the normalized root mean squared error (NRMSE) as

$$\text{NRMSE} = \sum_{i=1}^{q} \gamma_i \cdot \sqrt{\frac{\sum_{k=1}^{K}(y_i(k) - \hat{y}_i(k))^2}{\sum_{k=1}^{K}(y_i(k) - \text{E}(y_i))^2}} \tag{34}$$

where γ_i is the weighted factor for the i-th output that $\sum_{i=1}^{q} \gamma_i = 1$, and E stands for expectation.

The PC with CPU Intel(R) Core(TM) i5-6500 3.20 GHz and RAM 8.00GB is used for the simulation. The toolbox YALMIP in MATLAB is used for the optimization in the MPC. Simulation parameters are listed in Table I.

	M_{max}	G'	β	M	N_p	ρ
Case 1: Numerical plant	10	0.001	0.05	2	5	20
Case 2: Engine platform:	15	0.001	0.02	2	5	15

TABLE I: Simulation Parameters.

For comparison, the results for both the proposed LOLIMOT with the gradient-search adapted Gaussian parameters and the standard LOLIMOT ([18]), denoted as GS-LOLIMOT and LOLIMOT respectively, are shown in below figures.

A. Numerical plant

In this case, we consider the numerical plant [23] with two inputs and two outputs as

$$\begin{aligned} y_1(k+1) =& 0.4y_1(k) + \frac{u_1(k)}{1+u_1^2(k)} + 0.2u_1^2(k) + 0.5u_2(k) \\ y_2(k+1) =& 0.2y_2(k) + \frac{u_2(k)}{1+u_2^2(k)} + 0.4u_2^3(k) + 0.2u_1(k) \end{aligned} \tag{35}$$

1) Modelling performance of the numerical plant: There are 1200 sets of input-output pairs generated by the plant given by (35) with the input signal u_1 and u_2 being randomly set as 0 or 1. The first 1000 data sets are used for the model training and the remaining 200 data sets are used for testing.

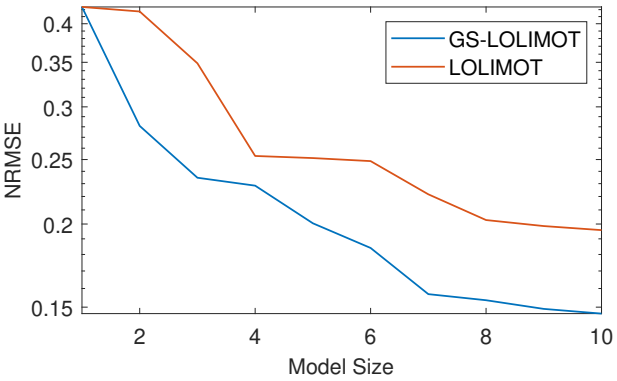


Fig. 5: Numerical plant training data: NRMSE *vs* model size

Fig. 5 and 6 show the NRMSE performance with respect to the model size for the training and test data sets, respectively. It is clearly shown that, for both GS-LOLIMOT and LOLIMOT, the NRMSE performance keeps improving with larger model size (i.e. the number of the neurons), but the performance improvement becomes less significant with large enough model size. For both the training and test data sets, the proposed GS-LOLIMOT has consistently better NRMSE performance than its LOLIMOT counterpart. On the other hand, for the same NRMSE, the proposed GS-LOLIMOT needs smaller model size than the LOLIMOT. For example, as shown in Fig. 6, to achieve NRMSE=0.2, the model sizes for the GS-LOLIMOT and LOLIMOT are about 4 and 8, respectively.

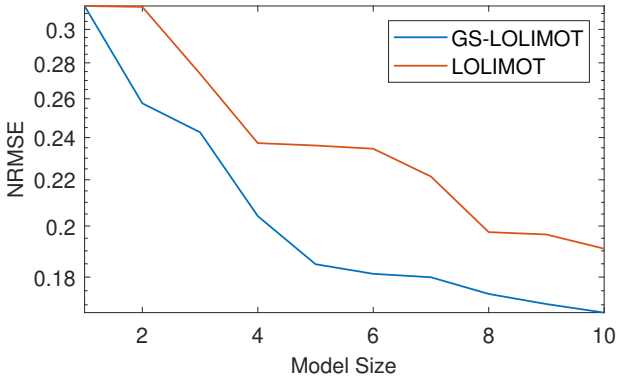


Fig. 6: Numerical plant test data: NRMSE *vs* model size

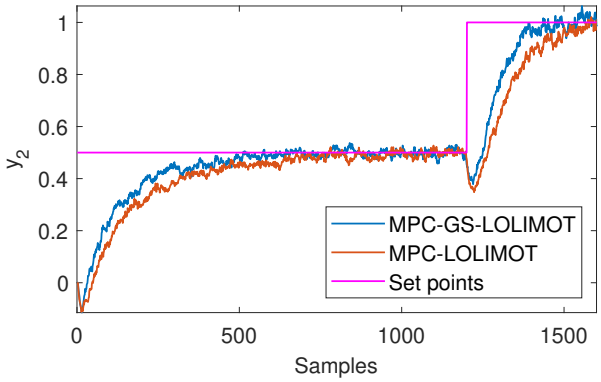


Fig. 8: Numerical plant: the tracking curves for y_2

Then in this case, the GS-LOLIMOT only needs half of the neurons to obtain the similar modelling performance of the standard LOLIMOT. This is particular important for the MPC because its complexity depends on the model size.

2) MPC performance of the numerical plant: The above trained LOLIMOT model with two neurons is used in the MPC, where the disturbances of SNR = 20 dB is added at the plant output, the prediction horizon N_p = 5, and the regularization factor in the cost function (29) ϱ = 20. The control performance for the MPC based on both the GS-LOLIMOT and LOLIMOT, denoted as MPC-GS-LOLIMOT and MPC-LOLIMOT respectively, are illustrated.

Fig. 7 and 8 compares the tracking curves for the MPC-GS-LOLIMOT and MPC-LOLIMOT for y_1 and y_2, respectively. While in both figures the MPC-GS-LOLIMOT can better track the setpoint variations than its the MPC-LOLIMOT counterpart, the improvement is more obviously for y_2 in Fig. 8. This is because controlling y_2 is more tolerate to model mismatch than y_1 in this example.

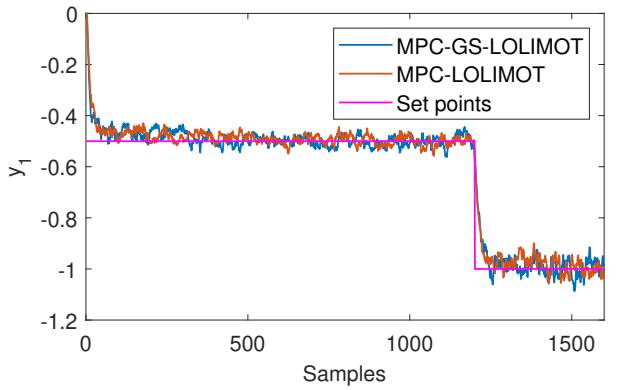


Fig. 7: Numerical plant: the tracking curves for y_1

B. Turbocharged diesel engine platform

The diagram of the engine platform is shown in Fig. 9. The turbocharged engine uses engine exhaust to improve the engine through the exhaust gas re-circulation (EGR) loop. At the exhaust manifold of the engine, part of the exhaust gas is recirculated directly back to the intake manifold of the engine through the EGR valve, and the rest of the exhaust drives the variable-geometry turbocharger (VGT) which again helps the compressor to produce higher pressured air to the engine intake manifold. By controlling the opening positions of the VGT vane and EGR valve, the required exhaust gas re-circulation mass flow rate (W_{egr}) and manifold air pressure (p_{in}) can be achieved.

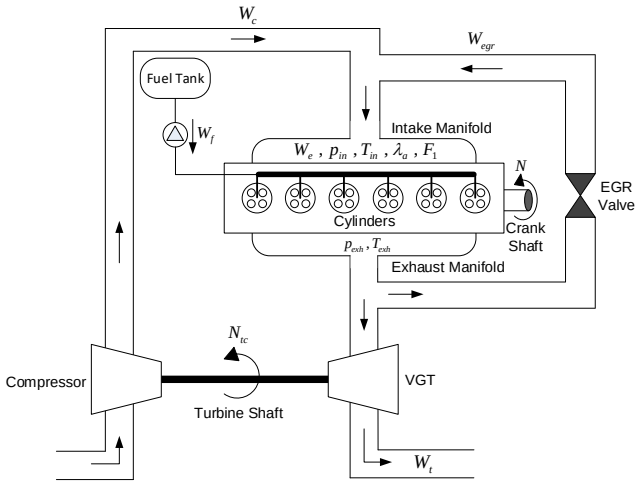


Fig. 9: Diagram of the turbocharged diesel engine

In this experiment, the plant inputs are the opening positions of the VGT vane and EGR valve, and the plant outputs are the W_{egr} and p_{in}. There is a strong coupling between the VGT vane and EGR valve because both are driven by the exhaust gas flow. On the other hand, the W_{egr} and p_{in} are also strongly coupled as both are affected by the VGT vane and EGR valve. With this consideration, the inputs and output of the neuron fuzzy model at time k (corresponding to Fig. 1) are given by $\boldsymbol{\xi}(k) = [\hat{p}_{\text{in}}(k), \hat{W}_{\text{egr}}(k), u_{\text{VGT}}(k), u_{\text{EGR}}(k)]^\top$ and $\hat{\boldsymbol{y}}(k+1) = [\hat{p}_{\text{in}}(k+1), \hat{W}_{\text{egr}}(k+1)]^\top$ respectively, where u_{VGT} and u_{EGR} are the opening positions of VGT vane and EGR valve, respectively.

1) Modelling performance of the engine plant: There are $10,000$ plant input-output data sets are generated by the turbocharged diesel engine simulator, with sampling frequency

100 Hz. The first 8000 data set are used to train the neuron-fuzzy model and the rest data set are for testing. Fig. 10 and

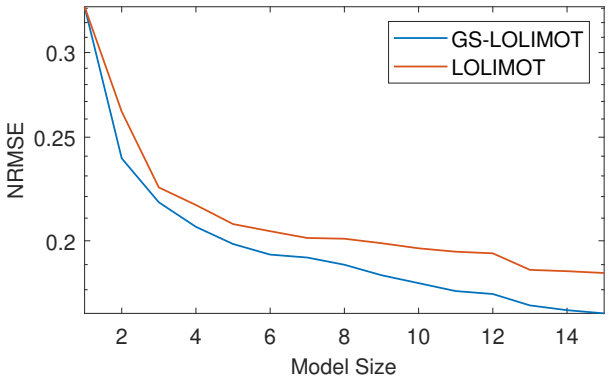


Fig. 10: Diesel engine plant training data: NRMSE vs model size

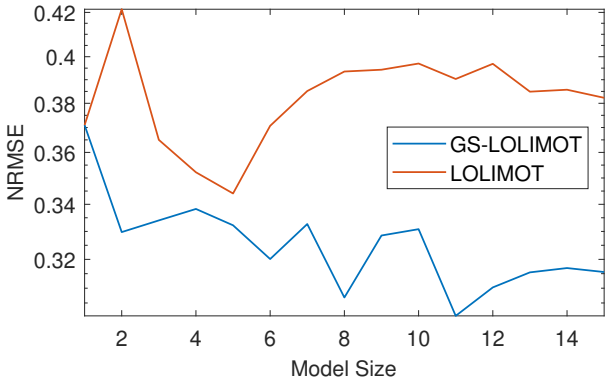


Fig. 11: Diesel engine plant test data: NRMSE vs model size

11 compare the modelling NRMSE with respect to the model size for the training and test data sets, respectively. For both training and test data, the GS-LOLIMOT achieves significantly lower NRMSE than the LOLIMOT. It is interesting to observe in Fig. 11 that the NRMSE stops decreasing when the model size reaches a certain value (about 5) due to overfitting. The results verify that, with adapted Gaussian parameters, the GS-LOLIMOT can better capture the underlying statistics of the plant data.

2) MPC performance of the engine plant: The above trained LOLIMOT with model size of two is applied to the MPC. In this experiment, the prediction horizon $N_p = 5$, and the regularization factor in the cost function (29) $\varrho = 15$.

Fig. 12 shows that the MPC-GS-LOLIMOT can significantly better track the setpoint for p_{in} than the MPC-LOLIMOT. For example, when the p_{in} setpoint varies from 180 kPa to 170 kPa at time 100s, it takes about 50 s and 100 s for the MPC-GS-LOLIMOT and MPC-LOLIMOT to converge, respectively. In other words, the proposed MPC-GS-LOLIMOT can track this setpoint variation 50 seconds faster than its MPC-LOLIMOT part. The improvement of the MPC-GS-LOLIMOT over the MPC-LOLIMOT is more obviously shown in Fig. 13 which shows the corresponding root-mean-square-error (RMSE) defined as $\sqrt{[r(k+1) - p_{\text{in}}(k+1)]^2}$. It is clearly shown that the MPC-GS-LOLIMOT converges not only faster but also to a lower RMSE.

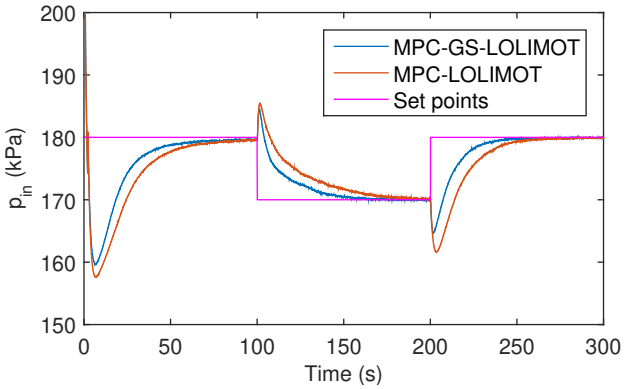


Fig. 12: Diesel engine plant: the p_{in} tracking curves

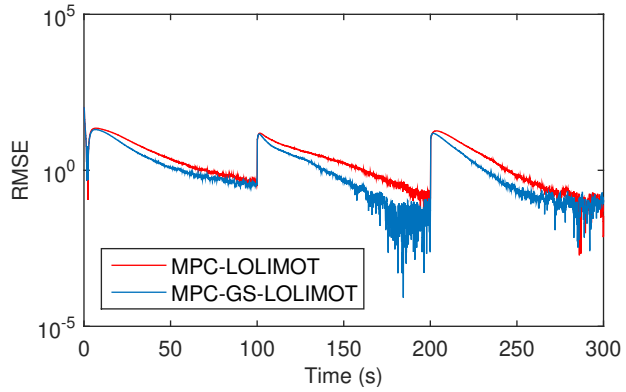


Fig. 13: Diesel engine plant: the p_{in} tracking RMSE-s performance

Fig. 14 show the tracking curves for W_{egr}, where the MPC-GS-LOLIMOT performs similarly, if not slightly better, than the GS-LOLIMOT. Thus for this plant, W_{egr} is more robust to model mismatch than p_{in}.

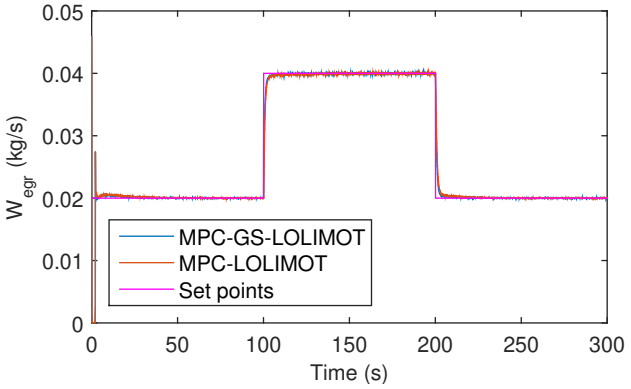


Fig. 14: Diesel engine plant: the W_{egr} tracking curves

Fig. 15 and 16 show the corresponding plant input curves for the VGT and EGR, respectively. The results also verifies that the proposed MPC-GS-LOLIMOT can better control the plant.

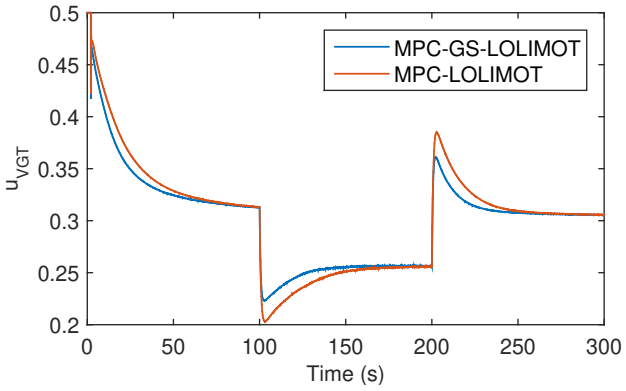


Fig. 15: Diesel engine plant: The VGT input curve

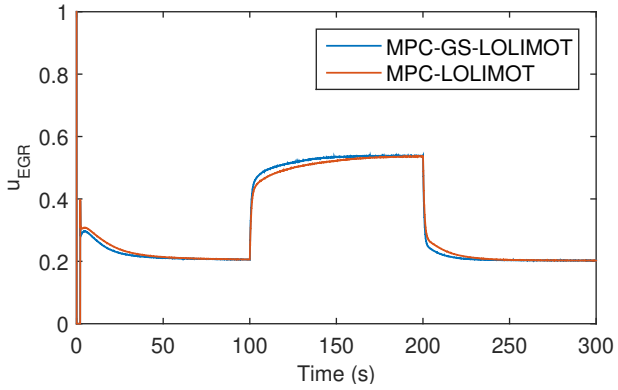


Fig. 16: Diesel engine plant: The EGR input curve

VI. Conclusion

This paper proposes a novel LOLIMOT with Gaussian parameters optimized by the gradient descent search approach, and successfully applies it in the MIMO MPC. Compared with the standard LOLIMOT, the structurally optimized LOLIMOT achieves lower RMSE with the same model size, or has smaller model size with the same RMSE. The MIMO-MPC based on the structurally optimized LOLIMOT can also better control the target plant. While the proposed LOLIMOT is trained offline based on the full batch of the training data, an interesting future work is to generalize the proposed work in the on-line case so that the LOLIMOT structure can track the variations of the underline system.

Appendix A
The Derivation of (32)

The i-th model output can be expressed as

$$\hat{y}_i(k+1) = [\widetilde{\boldsymbol{\omega}}^{(i)}]^\top \cdot \widetilde{\boldsymbol{\xi}}(k) \tag{36}$$

where $\widetilde{\boldsymbol{\omega}}^{(i)}$ is corresponding weight vector defined as $\widetilde{\boldsymbol{\omega}}^{(i)} = \sum_{m=1}^{M} \phi_m(k) \cdot \boldsymbol{\omega}_{m,i}$. From (36), the i-th model output is

$$\begin{aligned}&\hat{y}_i(k+n_p) = \hat{y}_i(k+n_p-1)+\\ &[\widetilde{\boldsymbol{\omega}}^{(i)}]^\top \cdot \widetilde{\boldsymbol{\xi}}(k+n_p-1) - [\widetilde{\boldsymbol{\omega}}^{(i)}]^\top \cdot \widetilde{\boldsymbol{\xi}}(k+n_p-1)\end{aligned} \tag{37}$$

Stacking (37) for all of the model outputs (from 1 to q) in a vector gives

$$\begin{aligned}&\begin{bmatrix}\hat{y}_1(k+n_p)\\ \vdots \\ \hat{y}_q(k+n_p)\end{bmatrix} = \bar{\boldsymbol{A}}_1 \cdot \begin{bmatrix}\hat{y}_1(k+n_p-1)\\ \vdots \\ \hat{y}_q(k+n_p-1)\end{bmatrix} + \\ &\bar{\boldsymbol{A}}_2 \cdot \begin{bmatrix}\hat{y}_1(k+n_p-2)\\ \vdots \\ \hat{y}_q(k+n_p-2)\end{bmatrix} + \cdots + \bar{\boldsymbol{A}}_{n_y^{\max}} \cdot \begin{bmatrix}\hat{y}_1(k+n_p-n_y^{\max})\\ \vdots \\ \hat{y}_q(k+n_p-n_y^{\max})\end{bmatrix} \\ &+ \bar{\boldsymbol{A}}_{n_y^{\max}+1} \cdot \begin{bmatrix}\hat{y}_1(k+n_p-n_y^{\max}-1)\\ \vdots \\ \hat{y}_q(k+n_p-n_y^{\max}-1)\end{bmatrix} + \bar{\boldsymbol{B}}_1 \cdot \\ &\begin{bmatrix}\Delta u_1(k+n_p-1)\\ \vdots \\ \Delta u_q(k+n_p-1)\end{bmatrix} + \cdots + \bar{\boldsymbol{B}}_{n_{u^{\max}}} \cdot \begin{bmatrix}\Delta u_1(k+n_p-n_u^{\max})\\ \vdots \\ \Delta u_q(k+n_p-n_u^{\max})\end{bmatrix}\end{aligned} \tag{38}$$

where $n_y^{\max}$ and $n_u^{\max}$ are the maximum lags for the corresponding output and input, respectively, and

$$\bar{\boldsymbol{A}}_1 = \begin{bmatrix}1-\widetilde{a}_{1,1}^{(1)} & 0-\widetilde{a}_{2,1}^{(1)} & \cdots & 0-\widetilde{a}_{q,1}^{(1)}\\ 0-\widetilde{a}_{1,1}^{(2)} & 1-\widetilde{a}_{2,1}^{(2)} & \cdots & 0-\widetilde{a}_{q,1}^{(2)}\\ \vdots & \vdots & \ddots & \vdots \\ 0-\widetilde{a}_{1,1}^{(q)} & 0-\widetilde{a}_{2,1}^{(q)} & \cdots & 1-\widetilde{a}_{q,1}^{(q)}\end{bmatrix}, \tag{39}$$

$$\bar{\boldsymbol{A}}_2 = \begin{bmatrix}\widetilde{a}_{1,1}^{(1)}-\widetilde{a}_{1,2}^{(1)} & \widetilde{a}_{2,1}^{(1)}-\widetilde{a}_{2,2}^{(1)} & \cdots & \widetilde{a}_{q,1}^{(1)}-\widetilde{a}_{q,2}^{(1)}\\ \widetilde{a}_{1,1}^{(2)}-\widetilde{a}_{1,2}^{(2)} & \widetilde{a}_{2,1}^{(2)}-\widetilde{a}_{2,2}^{(2)} & \cdots & \widetilde{a}_{q,1}^{(2)}-\widetilde{a}_{q,2}^{(2)}\\ \vdots & \vdots & \ddots & \vdots \\ \widetilde{a}_{1,1}^{(q)}-\widetilde{a}_{1,2}^{(q)} & \widetilde{a}_{2,1}^{(q)}-\widetilde{a}_{2,2}^{(q)} & \cdots & \widetilde{a}_{q,1}^{(q)}-\widetilde{a}_{q,2}^{(q)}\end{bmatrix} \cdots \tag{40}$$

$$\bar{\boldsymbol{A}}_{n_y^{\max}+1} = \begin{bmatrix}\widetilde{a}_{1,n_{y,1}}^{(1)} & \widetilde{a}_{2,n_{y,2}}^{(1)} & \cdots & \widetilde{a}_{q,n_{y,q}}^{(1)}\\ \widetilde{a}_{1,n_{y,1}}^{(2)} & \widetilde{a}_{2,n_{y,2}}^{(2)} & \cdots & \widetilde{a}_{q,n_{y,q}}^{(2)}\\ \vdots & \vdots & \vdots & \vdots \\ \widetilde{a}_{1,n_{y,1}}^{(q)} & \widetilde{a}_{2,n_{y,2}}^{(q)} & \cdots & \widetilde{a}_{q,n_{y,q}}^{(q)}\end{bmatrix}; \tag{41}$$

$$\bar{\boldsymbol{B}}_1 = \begin{bmatrix}\widetilde{b}_{1,1}^{(1)} & \widetilde{b}_{2,1}^{(1)} & \cdots & \widetilde{b}_{p,1}^{(1)}\\ \widetilde{b}_{1,1}^{(2)} & \widetilde{b}_{2,1}^{(2)} & \cdots & \widetilde{b}_{p,1}^{(2)}\\ \vdots & \vdots & \cdots & \vdots \\ \widetilde{b}_{1,1}^{(q)} & \widetilde{b}_{2,1}^{(q)} & \cdots & \widetilde{b}_{p,1}^{(q)}\end{bmatrix} \cdots \tag{42}$$

$$\bar{\boldsymbol{B}}_{n_{u^{\max}}} = \begin{bmatrix}\widetilde{b}_{1,n_{u,1}}^{(1)} & \widetilde{b}_{2,n_{u,2}}^{(1)} & \cdots & \widetilde{b}_{p,n_{u,p}}^{(1)}\\ \widetilde{b}_{1,n_{u,1}}^{(2)} & \widetilde{b}_{2,n_{u,2}}^{(2)} & \cdots & \widetilde{b}_{p,n_{u,p}}^{(2)}\\ \vdots & \vdots & \cdots & \vdots \\ \widetilde{b}_{1,n_{u,1}}^{(q)} & \widetilde{b}_{2,n_{u,2}}^{(q)} & \cdots & \widetilde{b}_{p,n_{u,p}}^{(q)}\end{bmatrix}. \tag{43}$$

Finally, from (38) and with some rearrangement, we can obtain (32), where $\boldsymbol{\Gamma}_1(k) = \boldsymbol{Q}^{-1}(k) \cdot \bar{\boldsymbol{\Gamma}}_{\mathbf{1}}(k)$, $\boldsymbol{\Gamma}_2(k) = \boldsymbol{Q}^{-1}(k) \cdot \bar{\boldsymbol{\Gamma}}_{\mathbf{2}}(k)$ and $\boldsymbol{\Gamma}_3(k) = \boldsymbol{Q}^{-1}(k) \cdot \bar{\boldsymbol{\Gamma}}_{\mathbf{3}}(k)$. and

$$\boldsymbol{Q}(k) = \begin{bmatrix} \boldsymbol{I}_q & 0 & 0 & \cdots & 0 \\ -\bar{\boldsymbol{A}}_1 & \boldsymbol{I}_q & 0 & \cdots & 0 \\ \vdots & -\bar{\boldsymbol{A}}_1 & \ddots & \cdots & \vdots \\ -\bar{\boldsymbol{A}}_{n_y^{\max}+1} & \vdots & \vdots & \boldsymbol{I}_q & 0 \\ 0 & 0 & \cdots & -\bar{\boldsymbol{A}}_1 & \boldsymbol{I}_q \end{bmatrix} \tag{44}$$

$$\bar{\boldsymbol{\Gamma}}_1(k) = \begin{bmatrix} \bar{\boldsymbol{A}}_1 & \bar{\boldsymbol{A}}_2 & \cdots & \bar{\boldsymbol{A}}_{n_y^{\max}} & \bar{\boldsymbol{A}}_{n_y^{\max}+1} \\ \bar{\boldsymbol{A}}_2 & \cdots & \bar{\boldsymbol{A}}_{n_y^{\max}} & \bar{\boldsymbol{A}}_{n_y^{\max}+1} & 0 \\ \bar{\boldsymbol{A}}_3 & \cdots & \bar{\boldsymbol{A}}_{n_y^{\max}+1} & 0 & 0 \\ \vdots & \vdots & 0 & \vdots & \vdots \\ \bar{\boldsymbol{A}}_{n_y^{\max}+1} & 0 & \vdots & \vdots & 0 \\ 0 & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{45}$$

$$\bar{\boldsymbol{\Gamma}}_2(k) = \begin{bmatrix} \bar{\boldsymbol{B}}_1 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ \bar{\boldsymbol{B}}_2 & \bar{\boldsymbol{B}}_1 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \bar{\boldsymbol{B}}_2 & \bar{\boldsymbol{B}}_1 & \cdots & 0 & \cdots & 0 \\ \bar{\boldsymbol{B}}_{n_u^{\max}} & \vdots & \vdots & \bar{\boldsymbol{B}}_1 & \vdots & \vdots & \vdots \\ \vdots & \bar{\boldsymbol{B}}_{n_u^{\max}} & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & \bar{\boldsymbol{B}}_{n_u^{\max}} & \cdots & \bar{\boldsymbol{B}}_2 & \bar{\boldsymbol{B}}_1 \end{bmatrix} \tag{46}$$

$$\bar{\boldsymbol{\Gamma}}_3(k) = \begin{bmatrix} \bar{\boldsymbol{B}}_2 & \bar{\boldsymbol{B}}_3 & \cdots & \cdots & \bar{\boldsymbol{B}}_{n_u^{\max}} \\ \bar{\boldsymbol{B}}_3 & \bar{\boldsymbol{B}}_4 & \cdots & \bar{\boldsymbol{B}}_{n_u^{\max}} & 0 \\ \bar{\boldsymbol{B}}_4 & \cdots & \bar{\boldsymbol{B}}_{n_u^{\max}} & 0 & 0 \\ \vdots & \vdots & 0 & \vdots & \vdots \\ \bar{\boldsymbol{B}}_{n_u^{\max}} & \vdots & \vdots & \vdots & \vdots \\ 0 & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{bmatrix} \tag{47}$$

and $\boldsymbol{I}_q$ is the q dimensional identity matrix.

REFERENCES

[1] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[2] B. Kouvaritakis and M. Cannon, "Model predictive control," *Switzerland: Springer International Publishing*, pp. 13–57, 2016.

[3] Y. Pan and J. Wang, "Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3089–3101, 2011.

[4] R. Hedjar, "Adaptive neural network model predictive control," *International Journal of Innovative Computing, Information and Control*, vol. 9, no. 3, pp. 1245–1257, 2013.

[5] L. Cheng, W. Liu, Z.-G. Hou, J. Yu, and M. Tan, "Neural-network-based nonlinear model predictive control for piezoelectric actuators," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7717–7727, 2015.

[6] A. S. Jamab and B. N. Araabi, "A learning algorithm for local linear neuro-fuzzy models with self-construction through merge & split," *2006 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 1–6, 2006.

[7] H. Peng, W. Wang, Q. An, C. Xiang, and L. Li, "Path tracking and direct yaw moment coordinated control based on robust MPC with the finite time horizon for autonomous independent-drive vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 6053–6066, 2020.

[8] Y. Dong, Y. Song, and G. Wei, "Efficient model predictive control for nonlinear systems in interval type-2 T-S fuzzy form under round-robin protocol," *IEEE Transactions on Fuzzy Systems*, 2020.

[9] E. Alcalá, V. Puig, and J. Quevedo, "TS-MPC for autonomous vehicles including a TS-MHE-UIO estimator," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 7, pp. 6403–6413, 2019.

[10] J. Fei and L. Liu, "Real-time nonlinear model predictive control of active power filter using self-feedback recurrent fuzzy neural network estimator," *IEEE Transactions on Industrial Electronics*, 2021.

[11] L. Cheng, W. Liu, Z.-G. Hou, T. Huang, J. Yu, and M. Tan, "An adaptive Takagi–Sugeno fuzzy model-based predictive controller for piezoelectric actuators," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 4, pp. 3048–3058, 2017.

[12] C. Sun, H. Gao, W. He, and Y. Yu, "Fuzzy neural network control of a flexible robotic manipulator using assumed mode method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5214–5227, 2018.

[13] R. Wai, J. Yao, and J. Lee, "Backstepping fuzzy-neural-network control design for hybrid maglev transportation system," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 302–317, 2015.

[14] A. Rubio-Solis and G. Panoutsos, "Interval type-2 radial basis function neural network: A modeling framework," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 457–473, 2015.

[15] X. Tang, L. Deng, and H. Qu, "Predictive control for networked interval type-2 T–S fuzzy system via an event-triggered dynamic output feedback scheme," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 8, pp. 1573–1586, 2019.

[16] M. Pratama, M. J. Er, X. Li, R. J. Oentaryo, E. Lughofer, and I. Arifin, "Data driven modeling based on dynamic parsimonious fuzzy neural network," *Neurocomputing*, vol. 110, pp. 18–28, 2013.

[17] Y. Y. Lin, J. Y. Chang, and C. T. Lin, "Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 2, pp. 310–321, 2013.

[18] 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.

[19] O. Nelles, "Local linear neuro-fuzzy models: fundamentals," in *Nonlinear system identification*, Springer Berlin / Heidelberg, 2000.

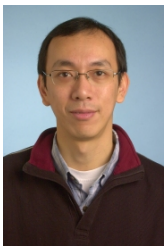
[20] R. Mehran, A. Fatehi, C. Lucas, and B. N. Araabi, "Particle swarm extension to LOLIMOT," in *Sixth International Conference on Intelligent Systems Design and Applications*, vol. 2, pp. 969–974, IEEE, 2006.

[21] R. Zhang and J. Tao, "A nonlinear fuzzy neural network modeling approach using an improved genetic algorithm," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5882–5892, 2017.

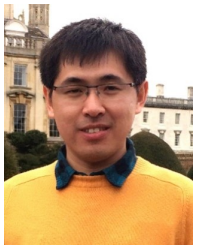
[22] M. Kvasnica and M. Fikar, "Design and implementation of model predictive control using multi-parametric toolbox and YALMIP," in *2010 IEEE International Symposium on Computer-Aided Control System Design*, pp. 999–1004, IEEE, 2010.

[23] Z. Yan, L. Weiwei, L. Xiuxia, and Y. Peng, "Output feedback control for discrete-time nonlinear systems and its applications," in *2009 Chinese Control and Decision Conference*, pp. 449–453, IEEE, 2009.

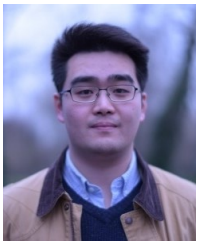
Xiaoyan Hu received her BEng degree in automatic control from Northeastern University, China, in 2015 and M.Sc. degree in computer science from Loughborough University, U.K., in 2017. She is currently pursuing the Ph.D. degree with the Wolfson School of Mechanical, Electrical and Manufacturing Engineering in Loughborough University, U.K. Her research interests are in the area of nonlinear system identification, machine learning, model optimization and fuzzy control.



Yu Gong has been with School of Electronic, Electrical and Systems Engineering, Loughborough University, UK, since 2012. Dr Gong obtained his BEng and MEng in electronic engineering in 1992 and 1995 respectively, both at the University of Electronics and Science Technology of China. In 2002, he received his PhD in communications from the National University of Singapore. After PhD graduation, he took several research positions in Institute of Inforcomm Research in Singapore and Queen's University of Belfast in the UK respectively. From 2006 and 2012, Dr Gong had been a lecturer in the School of Systems Engineering, University of Reading, UK. His research interests are in the area of signal processing and communications including wireless communications, cooperative networks, non-linear and non-stationary system identification and adaptive filters.



Dezong Zhao (M'12-SM'17) received the B.Eng. and M.S. degrees from Shandong University in 2003 and 2006, respectively, and the Ph.D. degree from Tsinghua University in 2010, all in Control Engineering. He was a Lecturer in Intelligent Systems at Loughborough University. He is currently a Senior Lecturer in Autonomous Systems with the University of Glasgow. His research interests include connected and autonomous vehicles, machine learning and control engineering. He has been an EPSRC Innovation Fellow since 2018 and a Royal Society-Newton Advanced Fellow since 2020.



Wen Gu received the M.S degree in automotive systems engineering in 2016 from the Loughborough University, Loughborough, UK, where he is working towards the Ph.D. degree in automotive engineering. He is presently a research and development engineer with the Department of Aeronautical and Automotive Engineering, Loughborough University and HORIBA-MIRA Ltd. His current research interests include nonlinear system identification, evolving systems and model-based calibration.