Data-driven multi-model control for a waste heat recovery system

Johan Peralez, Francesco Galuppo, Pascal Dufour, Christian Wolf, Madiha Nadri

Data-driven multi-model control for a waste heat recovery system

Johan Peralez[1], Francesco Galuppo[2], Pascal Dufour[1], Christian Wolf[3], Madiha Nadri[1*]

***Abstract*— We consider the problem of supervised learning of a multi-model based controller for non-linear systems. Selected multiple linear controllers are used for different operating points and combined with a local weighting scheme, whose weights are predicted by a deep neural network trained online. The network uses process and model outputs to drive the controller towards a suitable mixture of operating points.**

The proposed approach, which combines machine learning and classical control of linear processes, consists in the design of a controller for a waste heat recovery system (WHRS) mounted on a Heavy-Duty (HD) truck engine to decrease fuel consumption and meet the future pollutant emissions standard.

The proposed control scheme, which can be applied to any nonlinear system with an existing linear controller bank, is successfully evaluated on an Organic Rankine Cycle (ORC) process simulator and compared to a standard linear controller and to several strong multi-model baselines without learning.

I. Introduction

To describe the dynamic behavior of industrial processes, a nonlinear model is usually used. However the development of an accurate nonlinear first principle model can be difficult to obtain, as model equations and many parameters are usually hard to get from experimental results, and/or to use for identification purpose and control design. And even when it is possible to obtain such a model, the real-time and industrial constraints make its use often impossible for online control.

For these reasons, based on local models, conventional PID controllers have been the most used controllers due to their simplicity of design and their efficiency in industrial applications. On the downside, this approach is generally not satisfactory in presence of high non-linearities.

In this context, this work proposes a controller for non-linear systems based on multiple linearizations and a dynamic weighting scheme, with weights predicted by a deep neural network trained online in a supervised way. The neural network takes as input the process and model outputs as well as the errors over a short horizon, allowing it to learn their temporal evolution.

Related work — in the literature, first approximations of adaptive controllers were auto tuning methods [1]. We can also find different proposals based on different artificial neural networks that have been proposed for the self-tuning of PID controllers. In [2], the authors use a multilayer perceptron-neural network combined with a support vector machine for adaptive tuning of a PID controller. Reinforcement learning has also been used in the literature to design an adaptive PID based method, as in [3]. Compared to this work, we do not adapt the parameters of a single PID, but rather learn the dynamic weighting of an ensemble of PID controllers, which we argue to be better adapted to nonlinear systems.

Another approach to deal with nonlinear systems is the use of multiple linear controllers. That local controller can be combined with a weighting scheme as in [4] and [5] using a Bayesian estimator, or as [6] in which the control of a waste heat recovery system is addressed.

The process considered in this work — is an Organic Rankine Cycle (ORC) for recovering waste heat from a Heavy-Duty (HD) diesel engine. In HD transport the ORC technology is currently considered as a promising technology that could lead to essential reduction in fuel consumption [7]. The main differences with stationary applications lie in the highly transient behaviour of the hot source, depending on driving conditions. In this context, an effective control system is essential to attain satisfactory performance over a broad range of operating conditions. Recently the control of ORC systems have been intensively addressed and convincing results have been obtained such in [8], [9] where experimental results can be found.

However, the nonlinear controllers designed in this work highly depend on the quality of existing models. Unfortunately in ORC modeling, many physical characteristics must be known, such as fluid properties, evaporator geometry, transfer coefficients, whereas such data are often hard to come by. Therefore, our proposed approach offers an interesting alternative by the use of easy-to-obtain linear local models. In addition, the controller design is facilitated by the use of adaptive local linear controllers. Here, based on the online neural network estimator, we design a controller which combines feedback given by a "global" PID and a scheduled feedforward to track the superheat (SH) level at the inlet of the expander machine.

This paper is organized as follows: in Part II, a new control method with online tuning of the multi model weights based on deep neural network is presented. The bank of linear models is assumed to be already existing and their identification is not the scope of the paper. Part III deals with the application to an ORC simulator. The detailed model of the ORC is used here as a complete simulator only and is not used in the controller itself. From a driving cycle, the

[1] Univ Lyon, Université Claude Bernard Lyon 1, CNRS, LAGEPP UMR 5007, 43 boulevard du 11 novembre 1918, F-69100, Villeurbanne, France.

[2] The French ministry of higher education and research is acknowledged for the financial support of the CIFRE PhD thesis 2016/1205 of F. Galuppo between Volvo Trucks Saint Priest (France), Univ Lyon (France) and Univ Liège (Belgium).

[3] Université de Lyon, INSA-Lyon, LIRIS UMR CNRS 5205, Villeurbanne, France.

[*] Corresponding authors: Madiha Nadri (madiha.nadri-wolf@univ-lyon1.fr)

selection of the operating points for the design of the linear model bank and for identification is first briefly discussed. The proposed control scheme is then applied to this ORC and numerical results are discussed in Part IV.

II. Local multi model approach

As has been said in the introduction, multi-model approaches give an interesting alternative to circumvent the difficulties of single linearization points by taking into account several operating modes and designs in a time-varying global linear model. Several structures make it possible to interconnect the different local-models in order to generate the global output of the multi-model [4], [6]. In the specific context of Rankine cycle for transport applications, such an approach is also motivated by the wide range of operating conditions encountered during driving cycles.

For the general framework, let us consider a discrete nonlinear model of the form

$$\begin{cases} x(k+1) = f\big(x(k), u(k)\big) \\ y(k) = g\big(x(k), u(k)\big), \end{cases} \tag{1}$$

with $y \in \mathcal{R}^{n_y}$ the measured outputs, $u \in \mathcal{R}^{n_u}$ the measured inputs, and consider a set of local linear models

$$\begin{cases} x_i(k+1) = f_i\big(x_i(k), u(k)\big) \\ y_i(k) = g_i\big(x_i(k), u(k)\big) \end{cases} \tag{2}$$

each tuned to approximate (1) around an operating point. To obtain a global model, these local models have to be combined. The chosen approach is to construct a global model by linearly interpolating between the local models:

$$\hat{y}(k) = \sum_{i=1}^{N} w_i(k) y_i(k). \tag{3}$$

Here, the weights w_i provide a time-varying adaptation. They need to be identified on-line so that the outputs of the global model best match those of the plant. The considered problem is then to find an on-line estimation of the weights $w_i(k)$ such that the multi-model (2)-(3) approximates the initial system (1) on a wide range of operating conditions.

A. Related works

1) Bayesian estimator: One popular approach to determine the weight w_i is based on a Bayesian interpretation of the plant model mismatch [4], [5]. The recursive Bayesian weighting scheme is based on estimating the probabilities $p_i(k)$, each corresponding to the probability of the i^{th} model to be valid at step time t_k. Denoting $\epsilon_i(k) = y(k) - y_i(k)$ the i^{th} model error at time step t_k,

$$p_i(k) = \max\left(\delta, \frac{\exp(-0.5\epsilon_i(k) K \epsilon_i(k)^T) p_i(k-1)}{\sum_{j=1}^{N} \exp(-0.5\epsilon_j(k) K \epsilon_j(k)^T) p_j(k-1)}\right). \tag{4}$$

where δ and K are tuning parameters. A strictly positive value for δ keeps every model alive, while the K values govern the convergence speed toward a particular model. The weighting scheme is then computed with

$$\tilde{p}_i(k) = \begin{cases} p_i(k) & \text{for } p_i(k) > \delta \\ 0 & \text{else,} \end{cases} \quad w_i(k) = \frac{\tilde{p}_i(k)}{\sum_{j=1}^{N} \tilde{p}_j(k)}. \tag{5}$$

2) Distance based estimator: In [6] a simplified weighting scheme has been proposed to alleviate the difficulty in tuning the parameters of the Bayesian estimator. In this approach the modeling error is first normalized:

$$\tilde{\epsilon}_i(k) = \frac{\epsilon_i(k)^2}{\sum_{j=1}^{N} \epsilon_j(k)^2}, \tag{6}$$

$$c_i(k) = \big(1 - \tilde{\epsilon}_i(k)\big) \prod_{j \neq i, j=1}^{N} \tilde{\epsilon}_j(k). \tag{7}$$

The weighting scheme is then computed with

$$w_i(k) = \frac{c_i(k)}{\sum_{j=1}^{N} c_j(k)}. \tag{8}$$

B. Proposed neural network based estimator

In the present work we propose to use a deep neural network to estimate the weights w_i. At each time step t_k the network takes as input not only the last model errors $\epsilon(k)$ but the error sequences $\big(\epsilon_1(k), ..., \epsilon_1(k-d), ..., \epsilon_N(k), ..., \epsilon_N(k-d)\big)$ along with $\big(u(k), ..., u(k-d)\big)$, which provides it with information on the local evolution of the error. The motivation is to train the neural network to discriminate between the local models better suited to the encountered operating conditions.

1) Neural Network estimator: Let $z(k) \in \mathcal{R}^{(n_y \times N + n_u) \times d}$ be the flattened representation of the local modeling errors on the previous d sampling times concatenated with the process inputs:

$$\begin{aligned} z(k) = [&\epsilon_1(k), ..., \epsilon_1(k-d), ..., \epsilon_N(k), ..., \epsilon_N(k-d), \\ &u(k), ..., u(k-d)]. \end{aligned} \tag{9}$$

A deep neural network (DNN) is a non-linear function parameterized by a parameter vector Θ and is known to be able to approximate any continuous function on compact subsets $\mathbb{R}^N$ given a sufficient number of parameters [10], [11]. The network architecture corresponds to the analytical form of the parameterized functional mapping from the input $z(k)$ to the output $C(z(k))$, in our case it is composed of a sequence of linear layers each followed by a local non-linearity of type ReLU:

$$\begin{aligned} \widetilde{C}(z(k)) &= [\tilde{c}_1(z(k)), ..., \tilde{c}_N(z(k))] \\ &= L^p \phi(L^{p-1} \phi(L^{p-2} ... \phi(L^1 z(k)) ...)) \end{aligned} \tag{10}$$

$$C(z(k)) = \Phi(\widetilde{C}(z(k))), \tag{11}$$

where the activation function $\phi(.) = \max(., 0)$ is called the element-wise rectified linear unit (ReLU) function, and Φ is the softmax function:

$$\Phi(c_i(k)) = \frac{\exp(\tilde{c}_i(k))}{\sum_{j=1}^{N} \exp(\tilde{c}_j(k))}, \tag{12}$$

which ensures that the predicted outputs W_i are assigned values between 0 and 1 with a sum equal to 1.

The dimensions of matrices L are hyperparameters which govern the capacity of the neural model, i.e. the complexity of the relationships between variables it can represent — these hyperparameters are described in section IV-B; the last

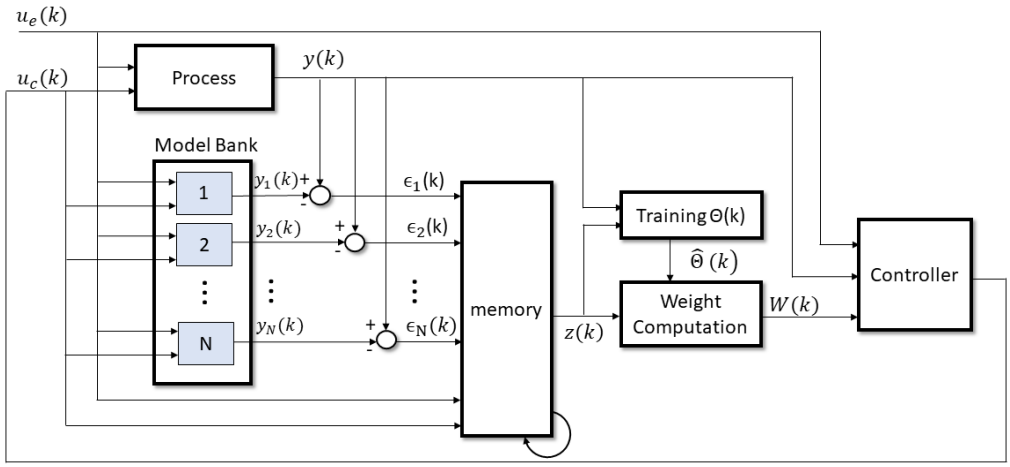


Fig. 1. Proposed multi-model controller: a bank of linear models combined with weights predicted by a deep neural network trained online. In the ORC application, the controlled input u_c correspond to the pump speed, the exogenous input u_e is the exhaust gas and y the superheating at evaporator outlet.

matrix is chosen such that the output C is of dimension N (= the number of linear models = the number of weights W_i to be predicted).

The predicted weights W_i for the different linear models are thus the outputs of the neural network:

$$W(k+1)=[w_1(k+1),\ldots,w_N(k+1)]=C(z(k)). \tag{13}$$

2) Online training: A classical method for supervised training of a neural network base estimator would be to sample input-output pairs $(z(k), y(k+1))$ by applying input values $z(k)$ to the process, measuring the process along with local-model outputs $y(k+1)$ and to train the neural network on the resulting data offline. The difficulty in this strategy is the generation of input sequences which lead to a proper exploration of the targeted operating conditions. Another weakness is a relatively low sample efficiency, resulting in the requirement of large amounts of training data to attain good performances over the targeted conditions.

We address these issues by training our neural network estimator online directly through interactions with the controller. As illustrated in Fig. 1, the weights W_i predicted by the network are used by the controller to generate the control signal towards the process and local-models. In return the neural network estimator receives process and models outputs that are used to improve the estimator parameters online.

During training, the parameters Θ of the neural network are updated to minimize a loss between measurements y and predicted $\hat{y}$, which, plugging Equation (3) into Equation (11) and making the dependency on network parameters Θ explicit, is given by

$$\hat{y}(k)=\textstyle\sum_{i=1}^{N} c_i(z_k,\Theta)y_i(k) \tag{14}$$

Given the continuous nature of the process output, we minimize the $\mathcal{L}_2$ loss given as

$$\mathcal{L}_2=(y-\hat{y})^2. \tag{15}$$

Inspired by similar work in reinforcement learning (see e.g [12]), we do not train the network on "hot" data directly estimated from the controller only. Instead, we keep a replay buffer of measurement sequences $(z(k), y(k+1))$ and sample this buffer randomly for training. This is known to reduce correlations in the data and to smooth changes over data distribution when the controller changes operating points.

We use stochastic gradient descent to iteratively update the network parameters Θ with samples from the current batch B of the replay buffer:

$$\Theta\leftarrow\Theta-\alpha\nabla_\Theta \textstyle\sum_{(y,\hat{y})\in B} \mathcal{L}_2(y,\hat{y}), \tag{16}$$

where α is the learning rate.

The neural network parameters may present fast fluctuations during the learning phase, especially during early stages. In consequence, the controller based on these estimations may present an erratic behavior. To alleviate this issue, and similar to work in the reinforcement learning literature [12], we propose the use of two networks during the learning phase. A first network with parameters Θ is continuously updated following gradient descent (16) while the parameters $\hat{\Theta}$ of a target network is updated periodically with Polyak's momentum:

$$\hat{\Theta}=\rho\hat{\Theta}+(1-\rho)\Theta, \tag{17}$$

where ρ is a hyperparameter between 0 and 1 (usually close to 1). Polyak's momentum prevents large parameter fluctuations of the target network (which provide weight estimations to the controller) without decreasing the learning rate [13].

III. Case study: Organic Rankine Cycle for transport application

ORC aims to recover heat from one or multiple heat sources to produce mechanical or electrical power. The heat is transferred to a working fluid (WF) that is selected according to the temperature levels of the heat sources and aspects related to safety, toxicity and impacts on the atmosphere. A key variable that has to be controlled to ensure the correct operation of the ORC is the superheat (SH) level at the inlet of the expander machine. It is defined as the

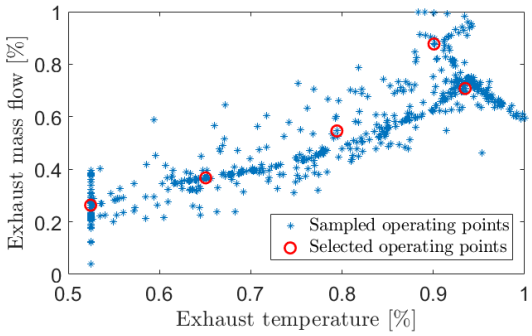


Fig. 2. Selection of operating points of the different linear models.

difference between the actual temperature of the WF at the inlet of the expander and the saturation temperature of the WF at its actual pressure at the inlet of the expander. In order to ensure safe operation of the expander, the SH must always be maintained at a strict positive value (consequently the expansion process can take place in the WF vapor state). For this set point tracking problem, the WF pump speed must be adapted continuously. Model based control of such a system is challenging, mostly due to complex heat exchanges with phase change behaviour of the working fluid and time varying conditions of the heat source due to the driving cycle in a HD truck.

A. Local linear models

1) Selection of operating points: Choosing the operating conditions for the identification of local models is an important step to obtain a global model effective on a wide range of operating conditions with a limited number of local models. To this end, real engine measurements have been collected from a French highway driving cycle (Lyon—Chambery—Grenoble) and stored in a dataset $D = \{x_j{=}(T_{exh,j}, \dot{m}_{exh,j})\}$ where T_{exh} is the exhaust gas temperature and $\dot{m}_{exh}$ the exhaust gas massflow. A subset $\hat{D}$ of N representative points is selected with respect to the following criterion:

$$\hat{D}{=}\arg\min_{D'} \sum_{x_j \in D} \min_{x'_j \in D'} ||x_j - x'_j||^2. \tag{18}$$

A genetic algorithm is used to perform the outer minimization in (18) — details are provided in section IV-B . The result of this procedure is illustrated in Fig. 2 for $N = 5$.

2) Identification of local models: Several previous studies on Rankine cycle processes showed that the relationship between pump actuator and superheating at evaporator outlet can be modeled by a low order linear system around an equilibrium point (see e.g [9]). In the present study a second order model is identified for each selected operating point, which can be written with the following transfer function:

$$\frac{y_u(p)}{u(p)} = \frac{K_u}{(1+\tau_{u,1}p)(1+\tau_{u,2}p)}, \tag{19}$$

where u is the pump speed and y_u the effect of pump speed on superheating.

Hereafter we further propose to model the relationship between the (non-controlled) exhaust gas flow and superheating at evaporator outlet. It was found that a first order transfer function was sufficient:

$$\frac{y_{exh}(p)}{\dot{m}_{exh}(p)} = \frac{K_{exh}}{1+\tau_{exh}p}, \tag{20}$$

where $\dot{m}_{exh}$ is the exhaust mass flow and y_{exh} the effect of exhaust flow on superheating. In the next section, (20) will allow the synthesis of a feedforward in the controller design.

Effective superheat at evaporator outlet is then $y = y_{exh} + y_u$. From the identification task based on pseudo-random binary sequence input, it is found that identified parameters vary drastically from one operating point to another. For example K_u is found to take values from -0.0659 to $-0.4160\,\mathrm{K\,rpm^{-1}}$ while τ_{exh} vary from 23.31 to $52.9\,\mathrm{s}$.

B. Control design

1) Scheduled PID Controller: As detailed in sec. III-A.2, input-output dynamics have been identified at different operating points via a set of linear models. The good fitting obtained with underdamped second-order transfer functions justifies the use of a gain-scheduled PID controller in the feedback path.

For each linear model a PID is tuned using the Internal Model Control (IMC) method where only one tuning parameter τ_m is required to govern the desired closed-loop time response. The PID for a local linear model i with identified parameters K_u, $\tau_{u,1}$, $\tau_{u,2}$ are then computed as follows. The Proportional, Integral and Derivative parameters — respectively denoted $K_{c,i}$, $K_{t,i}$, $K_{d,i}$ — are obtained using the IMC formulas:

$$\begin{aligned} K_{c,i} &= \frac{\tau u, 1 + \tau_{u,2}}{\tau_m K_{u,i}}, \; K_{t,i} = K_{c,i} \frac{1}{\tau_{u,1,i} + \tau_{u,2,i}} \\ K_{d,i} &= K_{c,i} \frac{\tau_{u,1,i}\tau_{u,2,i}}{\tau_{u,1,i} + \tau_{u,2,i}}. \end{aligned}$$

The neural network estimator developed in sec. II-B is then used to weight the local PIDs. The resulting global PID parameters are then $K_c = \sum_{i=1}^{N} w_i K_{c,i}$, $K_t = \sum_{i=1}^{N} w_i K_{t,i}$, $K_d = \sum_{i=1}^{N} w_i K_{d,i}$ and the feedback applied to the system is

$$u_{fb}(t) = K_c e(t) + K_t \int_0^t e(t)dt + K_d \frac{de(t)}{dt}, \tag{21}$$

$e(t) = y(t) - y^{SP}(t)$ is the error in tracking the setpoint.

2) Scheduled Feedforward: In the authors' previous paper [9], experimental results showed that a feedforward action can significantly improve the controller performances. Therefore a nonlinear model of the evaporator was inverted to be used in the feedforward path for disturbance rejection. This approach required an accurate first principle model on the fast transient conditions encountered during a real road driving cycle. In [9] this challenging issue is handled by the use of a so-called moving-boundaries model, governed by physical equations. Unfortunately many physical characteristics must be known, such as fluid properties, evaporator geometry, transfer coefficients, whereas such data are often hard to come by.

Hereafter a feedforward term is computed for each local model detailed in sec. II. Inspired by the approach in [9]

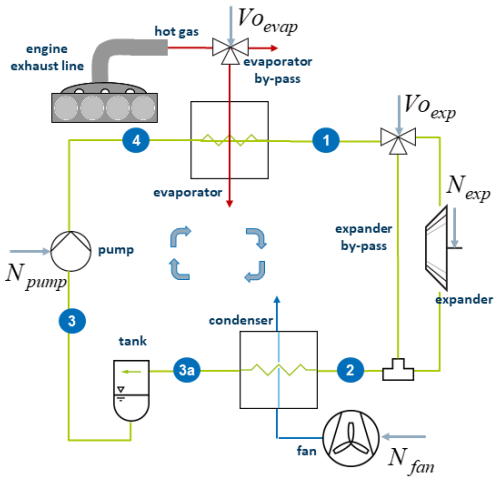


Fig. 3. ORC layout.

a two-time-scale dynamic behavior is assumed where (20) capture the slow dynamics. Neglecting $\tau_{u,1}$ and $\tau_{u,2}$, relation (19) is rewritten $u(p) = \frac{y_u(p)}{K_u}$. With (20), this yields

$$\begin{cases} \frac{y_{exh}(p)}{\dot{m}_{exh}(p)} = \frac{K_{exh}}{1+\tau_{exh}p} \\ u_(p) = \frac{y_u(p)}{K_u} \end{cases} \tag{22}$$

The feedforward part of the controller u_{ff} is computed as the value of u vanishing the effect of exhaust flow on superheating, i.e such that $y_u = -y_{exh}$. With (22), this gives

$$u_{ff}(p)=\frac{-K_{exh}}{K_u(1+\tau_{exh})}\dot{m}_{exh}(p) \tag{23}$$

Hence, each local model feedforward signal $u_{ff,i}(t)$ is computed from the exhaust gas signal $\dot{m}_{exh}$ filtered by the i^{th} first order of static gain $\frac{-K_{exh}}{K_u}$ and of constant time τ_{exh}. Notice that while the measurement $\dot{m}_{exh}$ is usually not directly available, a real-time estimation is provided by the engine control unit.

The neural network estimator is then used to weight the local feedforwards:

$$u_{ff}(k)=\sum_{i=1}^{N} w_i u_{ff,i}(k). \tag{24}$$

Finally feedback and feedforward signals defined in (21) and (24) are summed to give the control value sent to the pump actuator: $u(k) = u_{fb}(k) + u_{ff}(k)$.

IV. Simulation Results

A. Evaporator and ORC simulator

The system considered here is illustrated in Fig.3. It is an Organic Rankine Cycle (ORC) system for waste heat recovery from a HD Diesel engine using a volumetric expander. A high fidelity simulator of the ORC is implemented in Matlab-Simulink [14]. Although the present paper focuses on evaporator control, in order to prove the practical feasibility, the method is evaluated on the complete system. The evaporator is then subject to additional disturbances that are not considered in the local linear models detailed in II: the working fluid temperature at the evaporator inlet, the working fluid pressure and the exhaust gas temperature.

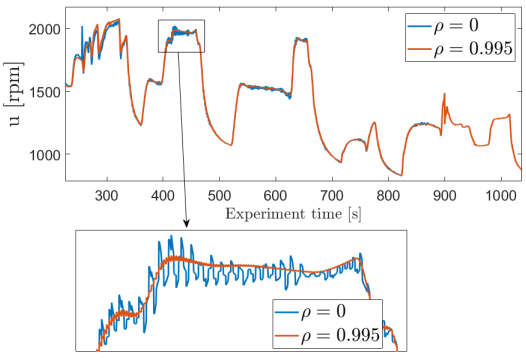


Fig. 4. Polyak benefits on actuator behavior during training stage: ρ close to one avoids pump speed fluctuations during early stages.

The linear model (19)-(20) could be augmented to take into account these disturbances. The main drawback would be an extensive additional identification campaign. The results presented hereafter show that the proposed method results in an effective regulation of superheating. Experiments are conducted on realistic conditions of HD road cycles.

B. Training

As the proposed control strategy is data driven and based on machine learning, we follow common practise and evaluate the approach on separate test data, which has not been seen during training, i.e. which has not been used for the gradient updates given in (16). In particular, two sets of data are used: one cycle containing 80,000 data samples for the learning stage and one cycle of equal size for testing.

The operation points are selected performing the minimization in (18) using the genetic algorithm implemented by Matlab's *ga* function with following parameters: a population of 200 samples, 10 elites, crossover fraction = 0.8, migration fraction = 0.2.

The neural network and its training are implemented in Python and using the Pytorch library. Sampling time for the weight computation are set to $0.5\,\mathrm{s}$, the neural network is fed with 10 previous samples (d=10) while 5 local linear models are used (N=5). In consequence, neural network inputs $z(k)$ — defined by (9) — are of dimension 70. The neural network (dense) structure is completed with two hidden layers of 64 neurons. Hence, using the notation of (10), the network architecture is given as p=4, $L^4 \in \mathcal{R}^{70\times64}$, $L^3 \in \mathcal{R}^{64\times64}$, $L^2 \in \mathcal{R}^{64\times64}$, $L^1 \in \mathcal{R}^{64\times5}$. The ADAM optimizer [15] was used for minimization with a batch size of 100.

The method detailed in II-B.2 is evaluated on the Matlab-Simulink ORC simulator. The proposed closed-loop strategy is found to succeed in training the neural network estimator with limited fluctuations induced by parameters updates. To illustrate the benefits of employing Polyak momentum, Fig. IV-B draws the controlled actuator behavior during early stages of learning. It compares the pump evolution with and without Polyak momentum (i.e for ρ respectively equal to 0.995 and 0): jumps in pump speed values are observed when Θ is updated. Polyak momentum prevents this issue while its use does not affect the training results in the long-term.

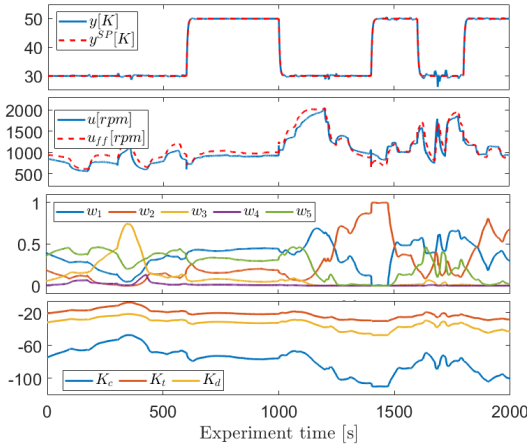


Fig. 5. Simulation results: superheating y (with setpoint y^{SP}), pump speed u (with feedforward part u_{ff}), weights w_i and resulting PID parameters K_c, K_i and K_d.

C. Closed loop control Results

Fig. 5 presents simulation results of the proposed approach on realistic driving conditions. The set point is changed during the simulation time to assess the controller performance. First, the proposed estimator for multi-model weights is evaluated. Simulations are conducted with each of the three estimators detailed in sec. II coupled with the same control strategy detailed in the preceding section. Performances in tracking output setpoint are compared in Table I. Results in Table I show interesting improvement in performances with both criteria ISE and IAE (improved by 35 to 40%.). However and despite the large inputs disturbances of the representative long haul truck driving cycle used, the superheating is kept close to the setpoint in the three experiments. Table II compares the two investigated variants of neural networks: the multi-layer perceptron given in equations (10) and recurrent neural networks (RNNs) in the long-short term memory (LSTM) variant [16]. There is no significant difference, suggesting that modelling long-term evolution of errors $\epsilon_i(k)$ is not crucial. Finally, to assess the benefit of the controller feedforward part in these results, the proposed estimator was used with only the feedback part of the controller (i.e with $u(k) = u_{fb}(k)$). A value of 0.2328 was obtained for ISE criterion (vs 0.0734 with feedforward) and 0.2416 for IAE criterion (vs 0.1376).

Weight estimator	**mean(ISE)**	**mean(IAE)**
Constant weights	0.2750	0.3025
Bayesian	0.1362	0.2101
Distance-based	0.1254	0.1898
Proposed neural network	**0.0734**	**0.1376**

TABLE I

PERFORMANCE COMPARISON WITH THE STATE OF THE ART

Weight estimator	**mean(ISE)**	**mean(IAE)**
Feed-forward Network (MLP)	0.0734	0.1376
Recurrent Network (LSTM)	0.0875	0.1495

TABLE II

SMALLINFLUENCE OF THE TYPE OF NEURAL NETWORK: HANDLING LONG-TERM TEMPORAL EVOLUTION (RNN) IS NOT CRUCIAL.

V. CONCLUSION

We proposed a multi-model based controller for non-linear systems, where the models are weighted by a neural network trained on a short past horizon. The network architecture is a multi-layer perceptron with ReLU activation functions and softmax normalization. Online training minimizes a loss between process output and the model output predicted by the weighted models, carried out through stochastic gradient descent. The approach is applied to an ORC simulator and compared with existing online weighting methods, illustrating that the SH setpoint tracking can be improved by 35 to 40%, hence leading to get further better ORC energy recovery.

REFERENCES

[1] K. Åström, T. Hägglund, C. Hang, and W. Ho, "Automatic tuning and adaptation for pid controllers - a survey," *Control Engineering Practice*, vol. 1, no. 4, pp. 699 – 714, 1993.

[2] F. G. Rossomando and C. M. Soria, "Identification and control of nonlinear dynamics of a mobile robot in discrete time using an adaptive technique based on neural pid," *Neural Computing and Applications*, vol. 26, pp. 1179–1191, 2015.

[3] E. O. Neftci and B. B. Averbeck, "Reinforcement learning in artificial and biological systems," *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133–143, 2019.

[4] N. N. Nandola and S. Bhartiya, "A multiple model approach for predictive control of nonlinear hybrid systems," *Journal of Process Control*, vol. 18, no. 2, pp. 131–148, 2008.

[5] B. Aufderheide and B. W. Bequette, "Extension of dynamic matrix control to multiple models," *Computers & Chemical Engineering*, vol. 27, no. 8-9, pp. 1079–1096, 2003.

[6] V. Grelet, P. Dufour, M. Nadri, V. Lemort, and T. Reiche, "Explicit multi-model predictive control of a waste heat rankine based system for heavy duty trucks," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 179–184.

[7] R. Daccord, A. Darmedru, and J. Melis, "Oil-Free Axial Piston Expander for Waste Heat Recovery," *SAE Technical Paper Series*, vol. 1, 2014.

[8] E. Feru, F. Kupper, C. Rojer, X. Seykens, F. Scappin, F. Willems, J. Smits, B. De Jager, and M. Steinbuch, "Experimental validation of a dynamic waste heat recovery system model for control purposes," SAE Technical Paper, Tech. Rep., 2013.

[9] J. Peralez, M. Nadri, P. Dufour, P. Tona, and A. Sciarretta, "Organic rankine cycle for vehicles: Control design and experimental results," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 3, pp. 952–965, 2016.

[10] G. Cybenko, "Approximations by superpositions of sigmoidal functions," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, p. 303–314, 1989.

[11] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The Expressive Power of Neural Networks: A View from the Width," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[14] F. Galuppo, T. Reiche, V. Lemort, P. Dufour, M. Nadri, and X. Huin, "Waste heat recovery (WHR) assessment in complete truck simulation environment," in *2018 SIA POWERTRAIN conference*, May 2018.

[15] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Machine Learning (ICML)*, 2015.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Data-driven multi-model control for a waste heat recovery system

Johan Peralez, Francesco Galuppo
Pascal Dufour, Christian Wolf, Madiha Nadri

IEEE Conference on Decision and Control, 2020

Motivation

- Approaches based on **Non-Linear models:** global results, hard to design.
- Approaches based on **Linear models**: local results, easier to design.
- Approaches based on **Multi-Linear models**: aim to obtain a global result by interpolation of a set of linear models/controllers.

Question: How can we design an efficient interpolation for a multi-model?

Our approach is based on learning from experiments.

Problem formulation

- Consider a discrete nonlinear model of the form:

$$\begin{cases} x(k+1) = f\big(x(k), u(k)\big) \\ y(k) = g\big(x(k), u(k)\big), \end{cases} \tag{1}$$

with $x \in \mathcal{R}^n$ the state, $y \in \mathcal{R}^{n_y}$ the measured outputs, $u \in \mathcal{R}^{n_u}$ the measured inputs.

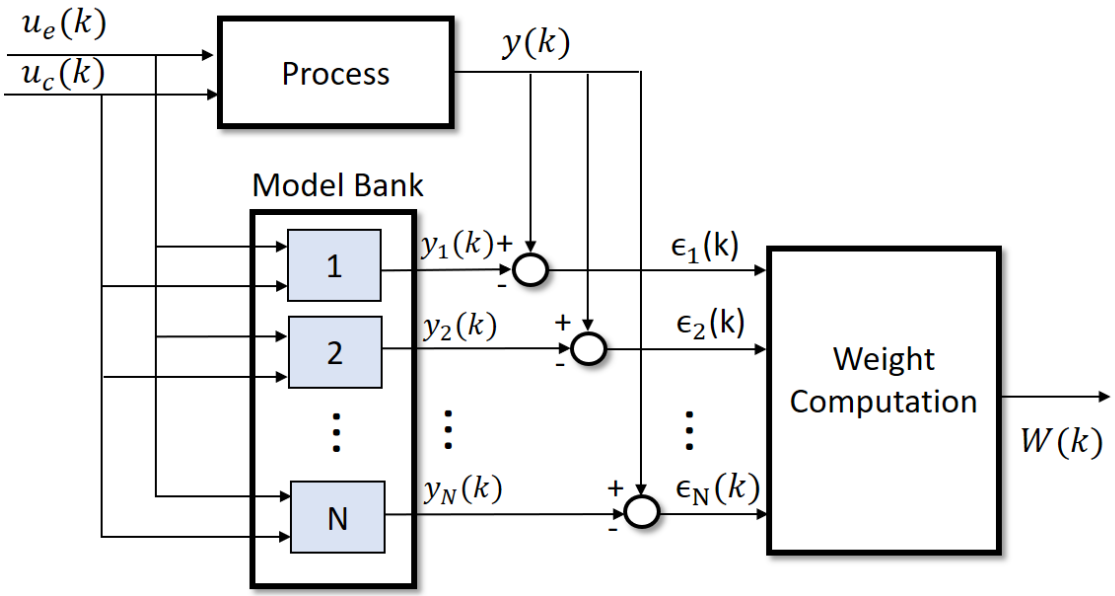
- Consider a set of local linear models:

$$\begin{cases} x_i(k+1) = A_i x_i(k) + Bu(k) \\ y_i(k) = C_i x_i(k) + D_i u(k) \end{cases} \tag{2}$$

- **Multi-model approach**: construct a global model interpolating between the local models:

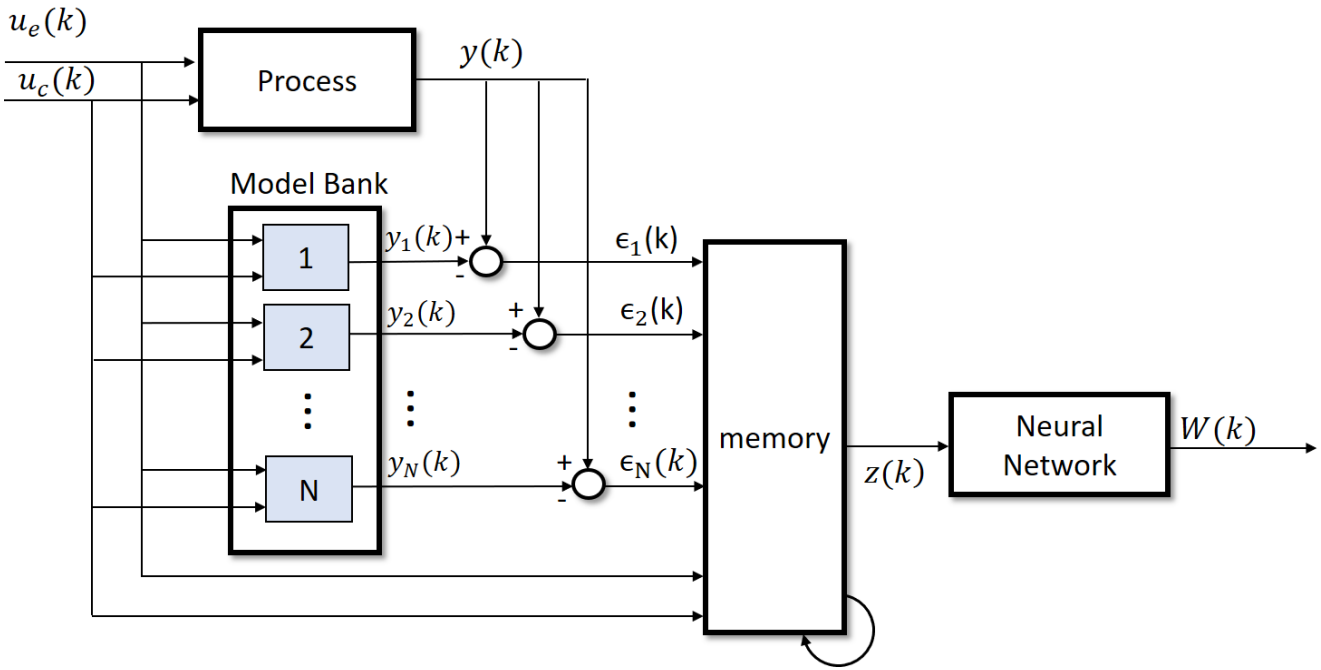
$$\hat{y}(k) = \sum_{i=1}^{N} w_i(k) y_i(k).$$

Related works



- **Distance based estimator:** weights $W(k)$ only depend on the prediction error $\epsilon(k)$.
- **Bayesian estimator:** weights also depend on previous errors (recursive scheme).

Proposed neural based estimator



- **Weights depend on prediction error and process input sequences:**

$$z(k) = [\epsilon_1(k), ..., \epsilon_1(k-d), ..., \epsilon_N(k), ..., \epsilon_N(k-d), u(k), ..., u(k-d), 1].$$

- We minimize the error between measurements y and predicted $\hat{y}$

$$\hat{y}(k) = \sum_{i=1}^{N} w_i(k) y_i(k)$$

Neural network structure

- **Multi Layer Perceptron** (MLP):

$$\begin{aligned}\widetilde{C}(z(k)) &= L^p\phi(L^{p-1}\phi(L^{p-2}...\phi(L^1 z(k))...)) \\ &= [\tilde{c}_1(z(k)), ..., \tilde{c}_N(z(k))],\end{aligned}$$

ReLU is chosen as the activation function: $\phi(.) = \max(., 0)$.

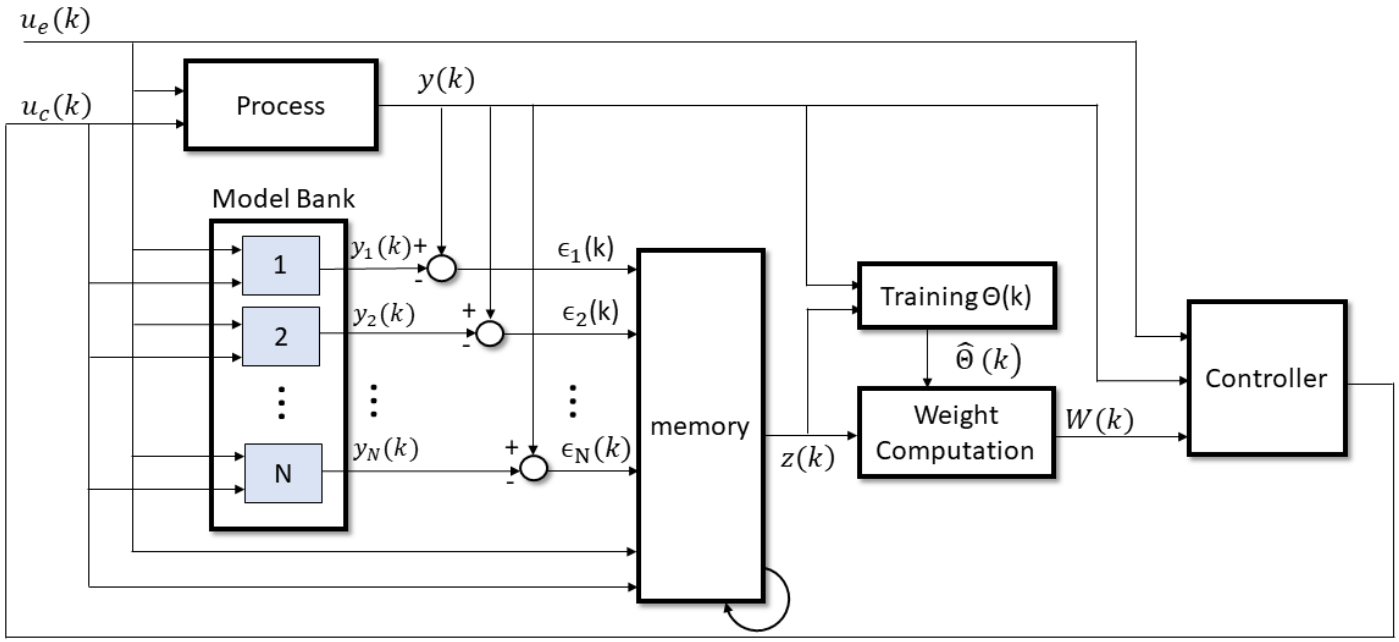
- With **Softmax** applied on outputs:

$$W(k+1) = \Phi(\widetilde{C}(z(k))),$$

Φ is the softmax function:

$$\Phi(c_i(k)) = \frac{\exp(\tilde{c}_i(k))}{\sum_{j=1}^{N} \exp(\tilde{c}_j(k))}.$$

Online training



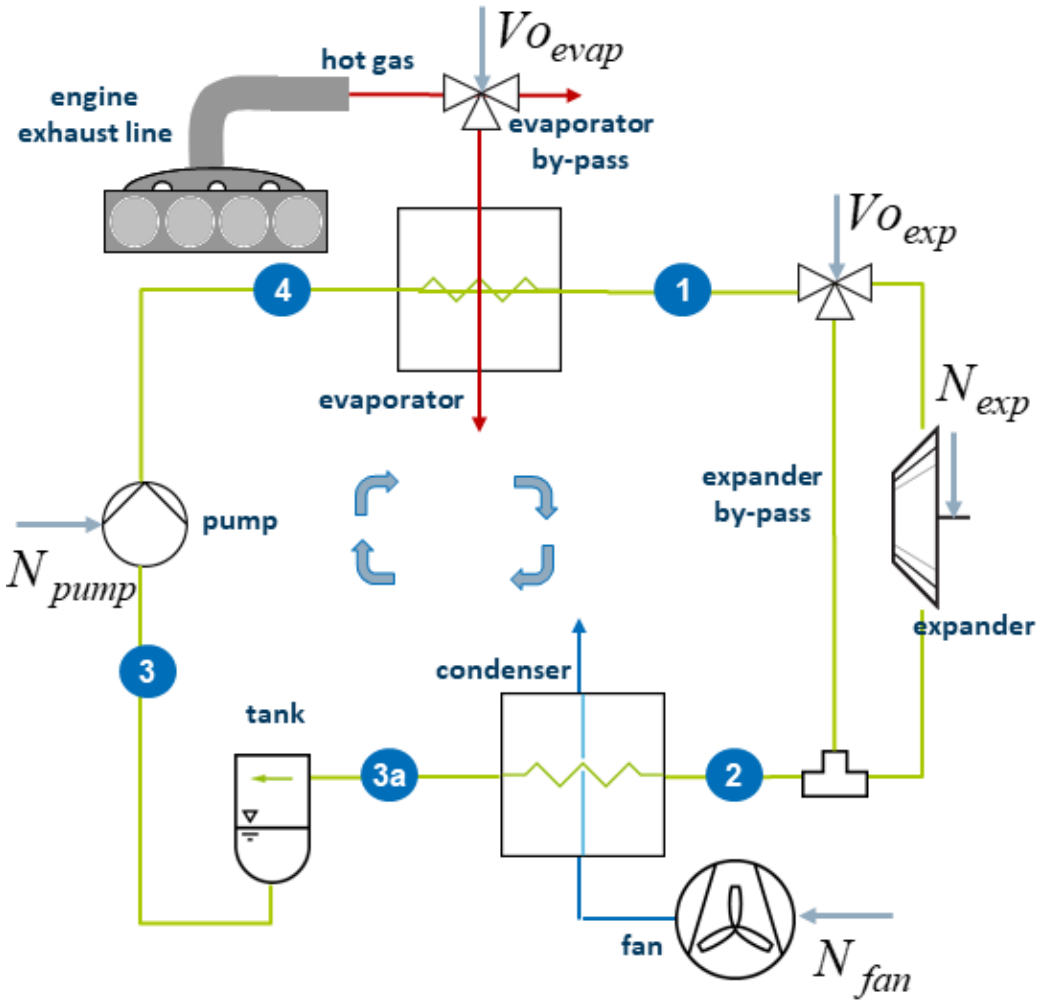
- **Supervised training**: minimize a loss between measurements and prediction.
- **Closed loop**: efficient exploration of the operating conditions.
- **A target network** prevents large parameter fluctuations:

$$\hat{\Theta}(k) = \rho\hat{\Theta}(k) + (1-\rho)\Theta(k),$$

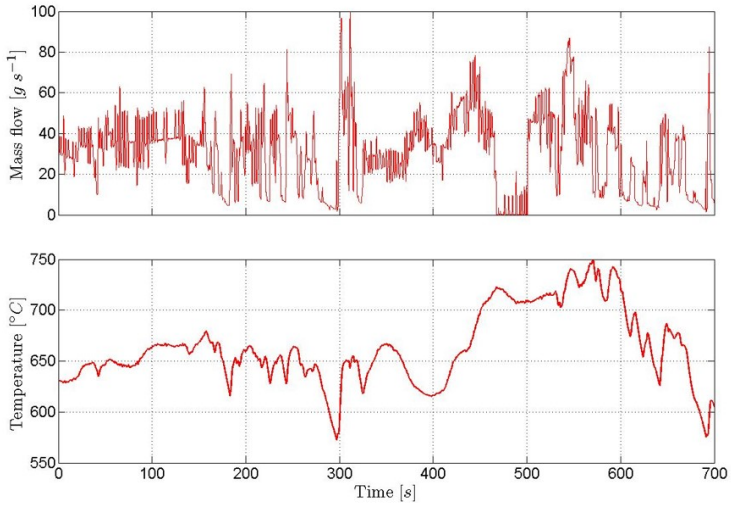
ρ is an hyperparameter $\in [0, 1]$ close to 1.

Case study: Rankine cycle

- Waste heat recovery for transport :



- Fast transient conditions :



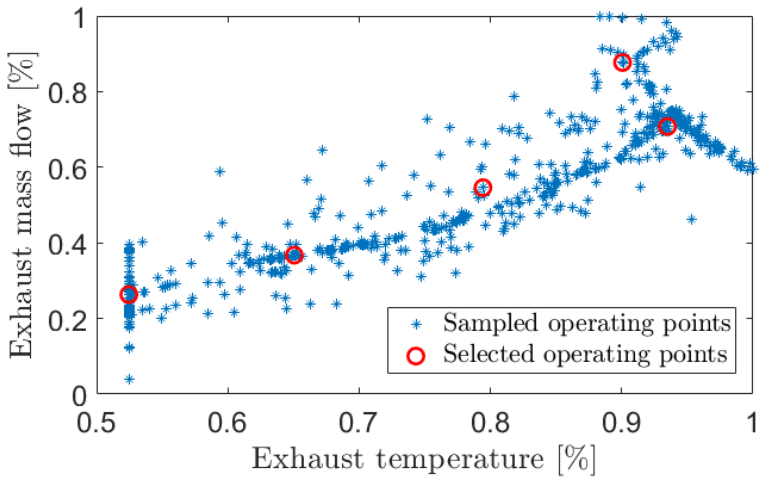
- We focus on the superheat (SH) control at the outlet of the evaporator.

Local linear models

- Pump-SH relationship:

$$\frac{y_u(p)}{u(p)} = \frac{K_u}{(1+\tau_{u,1}p)(1+\tau_{u,2}p)}.$$

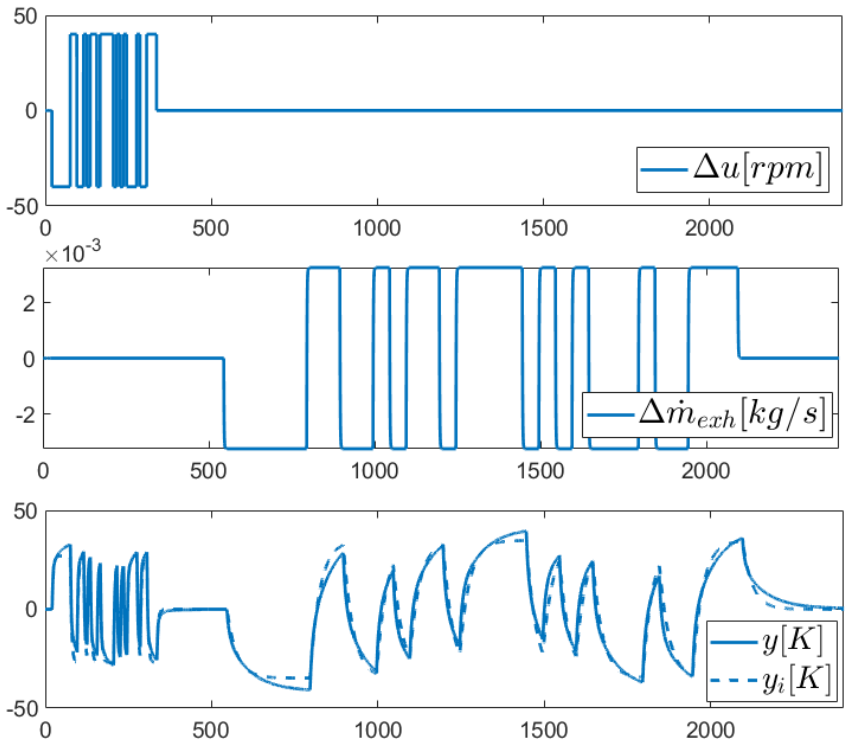
- Operating points selection:



- Exhaust-SH relationship:

$$\frac{y_{exh}(p)}{\dot{m}_{exh}(p)} = \frac{K_{exh}}{1+\tau_{exh}p}.$$

- Identification for an operating point:



Controller

- **Local PID controller**: tuned for each linear model (IMC method)

$$u_{fb,i}(k) = K_{c,i}e(k) + K_{t,i}\sum_{i=1}^{k} e(i) + K_{d,i}\frac{e(k) - e(k-1)}{\Delta t}.$$

- **Local Feedforward**:

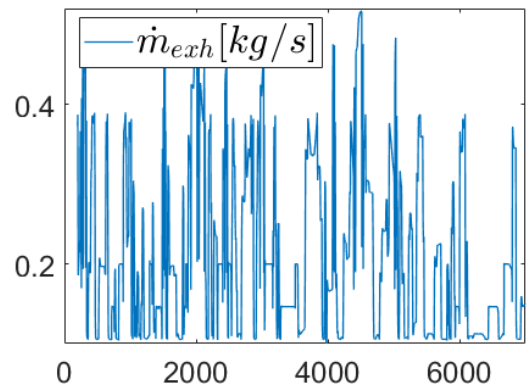
$$u_{ff,i}(k) = \frac{-K_{exh,i}}{K_{u,i}(1 + \tau_{exh,i})}\dot{m}_{exh}(k).$$

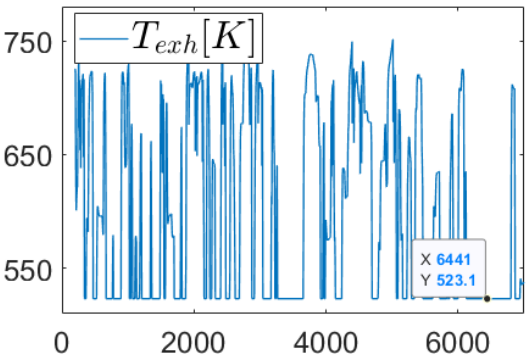
- **Scheduled controller**: the neural estimator weights the local controllers to have the global controller

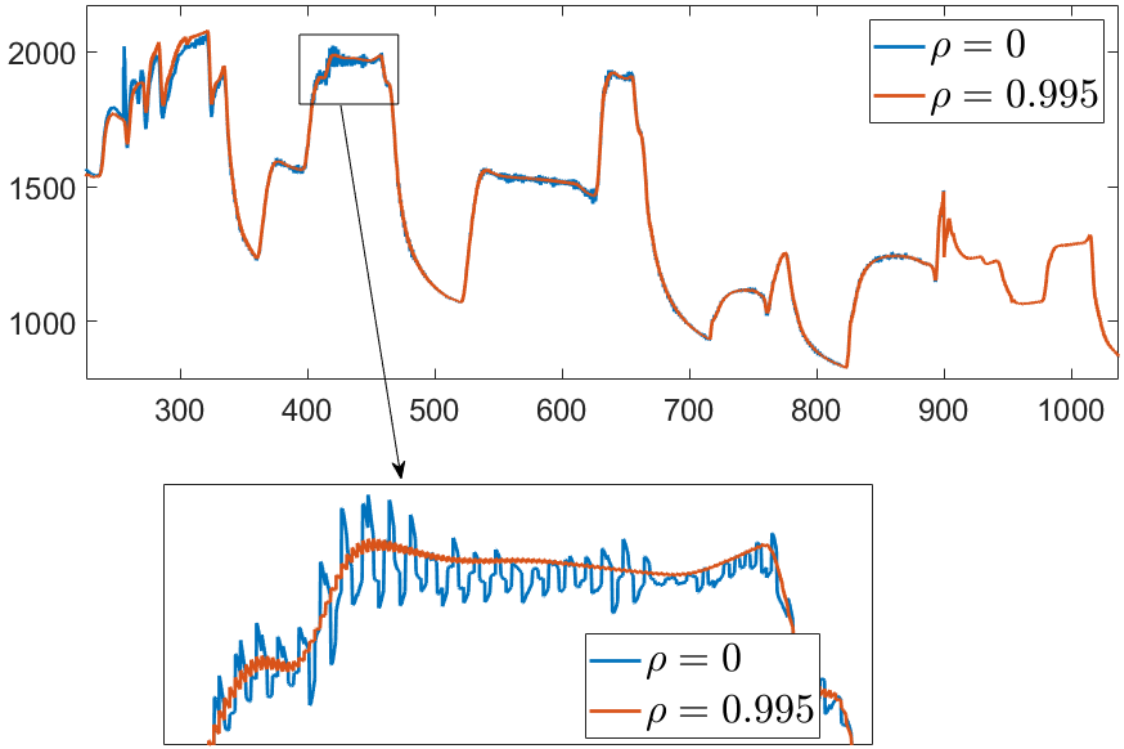
$$u(k) = \sum_{i=1}^{N} w_i(k)(u_{fb,i}(k) + u_{ff,i}(k)).$$

Online training results

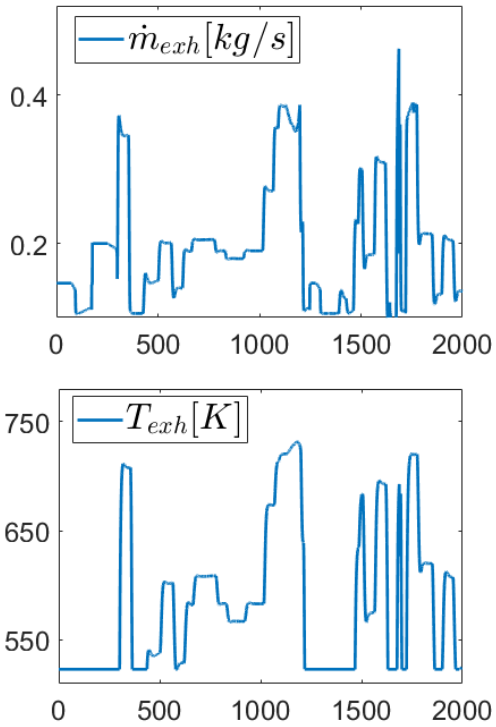
- 5 linear models
- Neural networks: two hidden layers of 64 neurons, fed with 10 previous samples ($d = 10$).
- Benefits of the target network on actuator fluctuation:

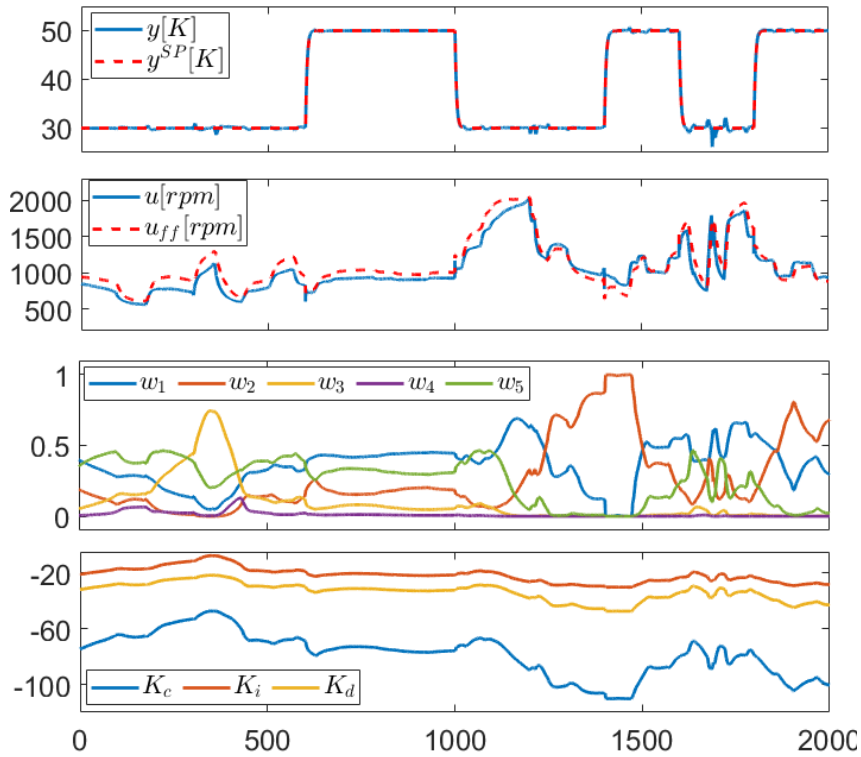






Controller performance





- Performance comparison:

Weight estimator	**mean(ISE)**	**mean(IAE)**
Constant weights	0.2750	0.3025
Bayesian	0.1362	0.2101
Distance-based	0.1254	0.1898
Proposed neural network	**0.0734**	**0.1376**

Conclusion

- A data driven multi-model based controller for non-linear systems is proposed.
- Local models are weighted by a neural network taking as input a short past horizon.
- Weights training is performed online through interactions.
- The approach is applied to an ORC simulator:
  - SH setpoint tracking can be improved by $35 - 40\%$
  - $=>$better ORC energy recover, CO_2 emission reduction.

Future work:

- Use higher number of linear models.
- Dynamic activations and deactivations, targeting high expressivity while keeping low computational cost.