ResNet and PolyNet based identification and (MPC) control of dynamical systems: a promising way

Pierre Clement Blaud, Philippe Chevrel, Fabien Claveau, Pierrick Haurant, Anthony Mouraud

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier

ResNet and PolyNet based identification and (MPC) control of dynamical systems: a promising way

PIERRE CLÉMENT BLAUD[1], PHILIPPE CHEVREL[2], FABIEN CLAVEAU[3], PIERRICK HAURANT[4], ANTHONY MOURAUD[5]

[1]CEA, CEA Tech Pays de la Loire, F-44340 Bouguenais and IMT Atlantique, LS2N, UMR CNRS 6004, F-44307 Nantes, France. (e-mail: pierre-clement.blaud@imt-atlantique.fr)
[2]IMT Atlantique, LS2N, UMR CNRS 6004, F-44307 Nantes, France. (e-mail: philippe.chevrel@imt-atlantique.fr)
[3]IMT Atlantique, LS2N, UMR CNRS 6004, F-44307 Nantes, France. (e-mail: fabien.claveau@imt-atlantique.fr)
[4]IMT Atlantique, GEPEA, UMR CNRS 6144, F-44307 Nantes, France. (e-mail: pierrick.haurant@imt-atlantique.fr)
[5]CEA, CEA Tech Pays de la Loire, F-44340 Bouguenais (e-mail: anthony.mouraud@cea.fr)

Corresponding author: Pierre Clément Blaud (e-mail: pierre-clement.blaud@imt-atlantique.fr).

"This research was funded by Conseil Régional des Pays de la Loire, France."

ABSTRACT This paper deals with model predictive control synthesis which take benefits from artificial neural networks to model (non-linear) dynamical system. More precisely, thanks to a systematic and rigorous methodology, it is shown that residual networks (ResNet) and PolyInception networks (PolyNet) neural network architectures, developed initially for image recognition, are very good candidate for i) identification of dynamical systems, ii) being used as embedded model in a model predictive control laws. Concretely, the widely used non-linear dynamical system quadruple tank process is used as a benchmark. The neural network architectures studied are i) feedforward networks as a reference point, and the two other linked to Euler integration method ii) residual networks and iii) PolyInception networks. Networks training is performed by mixing classical back-propagation algorithm and hyperparameters optimisation through heuristics. The identification results provided show that neural networks of types ii) and iii) perform better than the classical one i), with a better generalisation capability. Finally, model predictive controllers are synthesized based on the various networks trained. The simulation results obtained for controlling water levels of a 4 tanks system benchmark give interesting insights. They show that residual networks based model predictive control is better suited than feedforward networks and PolyInception networks based ones, both taking into account computation time and set point errors.

INDEX TERMS Artificial neural networks, model predictive control, feedforward neural networks, ResNet, PolyNet, quadruple tank process.

I. INTRODUCTION

AMONGST advanced control techniques, Model Predictive Control (MPC) has been commonly used in academia [1] and industrial applications [2]. MPC can control multi-inputs, multi-outputs dynamical systems while handling state and input constraints. The main parts of MPC are the cost function, dynamical system model and constraints [3]. In classical MPC, the dynamical system is represented using a state-space model which can be linear [4] or non-linear [5]. The cost function is optimised online, and the first sample of the control law computed over the receding horizon is applied to the actuators.

One concern about MPC regards finding a suitable discrete model of the non-linear dynamical system to control [6] which predicts the physical phenomena of a process [7]. This model is usually derived from discretised continuous Differential Equations (DEs) [8]. The discretisation is performed using an integration method such as by Euler [9], Runge-Kutta [10], Bogacki-Shampine [11], or Adams-Bashforth [12]. Furthermore, finding DEs could be labour intensive, and it could be advantageous to use measure data when available to identify a model [13], because nowadays an increasing amount of process data are available [14].

Machine learning is used to learn features from data, such as to identity objects from images, match items, transcribe speech or choose relevant search results. A common method

is to learn the feature with supervised machine learning, where the purpose is to learn a function that can map the input and output pairs with the highest score [15], [16]. In control systems, system identification usually uses linear parametric models from measured data [17]. However, it has been shown that Artificial Neural Network (ANN) can outperform linear parametric models and reduce identification error [18]. Supervised learning has been used to identify complex non-linear dynamical systems [19], [20], and ANN have been assessed and evaluated for dynamical system identification. Without claiming to be exhaustive, we can cite FeedForward Neural Network (FNN) [21], [22], Time Delay Neural Network (TDNN) [23], [24], Recurrent Neural Network (RNN) [25]–[27], modern RNN with gating mechanisms such as Long Short Term Memory (LSTM) [18], [28], [29] or Gate Recurrent Unit (GRU) [30], [31].

The relation between numerical integration methods and ad hoc neural network architectures has been recently observed in [32] and other works [33], [34].

In [32] and [34], the authors noted the link between the Residual layer Network (ResNet) [35] and the forward Euler integration method [9], while [33] depicts the link between PolyInceptions neural Network (PolyNet) [36] and the backward Euler integration method [9]. A neural network architecture derived from higher order integration method is described as a Runge-Kutta neural network [37], which are also considered in [38]. There is a major difference between neural networks derived from integration methods such as ResNet and others such as FNN. In the first group, the rate of change is learned, while this is the state in the second group [37], [38].

Dynamical systems in control engineering could be considered as a control benchmark, which are not a fully industrial plant but more a plain toy process. These systems allow highlighting, developing, testing and comparing control methods or algorithms before considering industrial applications in a more sophisticated plant. An example is the inverted pendulum on a cart, which is well known by every graduate student in control engineering for learning feedback control [39]. Another example is the four tanks, known as Quadruple Tank Process (QTP) [40], which has been used to evaluate predictive control in [41], robust economic MPC in [42], sliding model control in [43] or proportional–integral–derivative auto-tuners from manufacturers in [44]. The continuously stirred tank reactor is another benchmark in chemical engineering processes [45], which is used to present a two-layer EMPC control [46], a robust nonlinear MPC control [47] or an MPC approximated by a neural network [48]. Another example is the blood glucose control relevant model for type-I diabetic [49], with a robust MPC control approach in [50] or contractive MPC in [51]. Another example is the spacecraft, which has been use to evaluate receding horizon control in [52] and robust MPC in [53]. In this work, we chose the QTP to comparatively evaluate candidate neural network architectures and the associated MPC implementation, because the QTP is both simple and popular in the MPC community [41], [42], and its behaviour is non-linear despite its simplicity. The QTP is also naturally linked to issues arising from the processing industry, such as wastewater treatment, nuclear steam generation, chemical treatment or food processing [54].

Finally, our contribution regards recent work aiming to control the QTP with an ANN using an FNN in [54] and RNN in [55]. These works use FNN and RNN, while our objective is to illustrate using a dedicated architecture which is articulated with the digital integration process.

Although many ANNs have been used for dynamic system identification, the recent discovery of architectures better suited to digital integration needs has not led to precise numerical comparisons. The main contribution of this work is to present an accurate comparison of ResNet and PoyNet and feedforward shallow architectures regarding their ability to identify dynamical systems in terms of computational complexity and forecast accuracy. The aim is to provide a numerical illustration using the case studies' QTP, which represents important industrial problems. An MPC command will be thereby developed for the QTP using input-output data via neural identification of the process model. Three MPCs are considered: one with FNN, the second with ResNet and the third with the PolyNet model. As a result, it leads to the FNN-MPC, ResNet-MPC and PolyNet-MPC. To the author best knowledge, this is the first time where neural networks link to integration method are used as a model for an MPC in order to control a dynamical system. The contributions of the present work can be divided into three main elements:

- A presentation of different neural network architectures for non-linear dynamical system identification with supervised machine learning;
- A fair comparison between the three neural networks is considered in this work;
- A predictive controller based on neural models to control the process considered in this work.

This paper is organised as follows: Section II presents a system setup; Section III depicts neural networks; Section IV presents neural computations; Section V presents the MPC; Section VI depicts the experimental setup; Section VII presents the results; Section VIII depicts the discussions of the work; and Section IX concludes the work.

II. DYNAMICAL SYSTEM IDENTIFICATION

The first step is to define by (1) the discrete time invariant system considered below:

$$\bar{x}[k+1] = f(\bar{x}[k], \bar{u}[k]) \tag{1}$$

with $\bar{x}[k]$ and $\bar{u}[k]$ as the system state and input respectively, k the discrete sampling time and $\bar{x}[k] = \bar{x}(t_k)$, $t_k = k \times T_s$. In addition, T_s is the sample time. The absence of an output equation is linked to the assumption that all the states are assumed to be known and are therefore usable by a possible

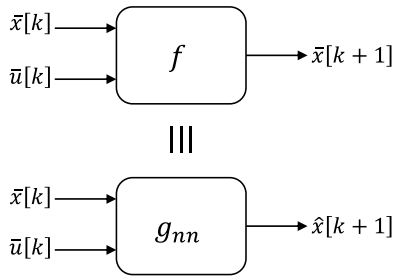

FIGURE 1. Model input-output representation, with f as the truth unreachable dynamical system and g_{nn} a neural network model.

control law. The objective is to learn from data an appropriate approximation of the nonlinear system (1) of the form:

$$\hat{x}[k+1] = g_{nn}(\bar{x}[k], \bar{u}[k]) \tag{2}$$

with $\hat{x}[k+1]$ as the predicted state of $\bar{x}[k+1]$ and g_{nn} the state evolution function of the predictor, which is expected to be close as possible to f, according to Fig. 1. Finding the nature of the relation linking $(\bar{x}[k], \bar{u}[k])$ and $\bar{x}[k+1]$ is unreachable by science (f does not formally exist for a physical system), and it is only an approximation [56].

III. NEURAL NETWORK

The notation used for the neuron units are h for the output of a neural unit at sample instant k and v for the input.

A. ARTIFICIAL NEURON

The first artificial neuron was developed by McCulloh and Pitts [57]. In this work, the artificial neuron considered is defined as [58]:

$$h = \sigma(Wv + b) \tag{3}$$

with h as the output of the cell, σ the activation function, W the weighting vector, v the input of the cell and b the bias.

B. FEEDFORWARD NEURAL NETWORK

When artificial neurons are combined, this leads to a networks of neurons, which is formalised by [58]:

$$h = \mathcal{G}(v) := g_{h^1}(v) \odot g_{h^2}(h^1) \odot \ldots \odot g_{h^{i-1}}(h^{i-2}) \odot g_{h^i}(h^{i-1}) \tag{4}$$

with h as the output of the FNN and v the input, g_{h^i} the non-linear transformation of hidden layer ^{i}, $\odot$ the Hadamard (elementwise) product operator. At the hidden layer ^{i}, g_{h^i} is $h^i = g_{h^i}(h^{i-1}) = \sigma^i(W^i h^{i-1} + b^i)$, h^i is the output, h^{i-1} is the input, σ^i is the activation function, b^i is the bias and W^i the weighting matrix. Note that layers 1 and $i-1$ from (4) are not hidden layers, but rather they are input and output layers (merged if only one layer is used in the network).

C. RESNET

Residual layer networks have been proposed in [35] to propagate the residual value of the input along the hidden layers in order to avoid vanishing gradient issues when training deep neural networks. The main feature of the ResNet is a skip connection and addition between each cell. ResNet is defined as:

$$h = v + \mathcal{G}(v) = (I + \mathcal{G})(v) \tag{5}$$

with h as the output and v the input, while $\mathcal{G}$ is an FNN as in (4).

D. POLYNET

PolyInceptions neural networks were proposed in [36] for image recognition, and they have multiple paths and lead to polynomial compositions. PolyNet is defined as:

$$h = v + \mathcal{G}(v) + \mathcal{G}(\mathcal{G}(v)) = (I + \mathcal{G} + \mathcal{G}^2)(v) \tag{6}$$

with h as the output and v the input, while $\mathcal{G}$ is an FNN as in (4).

E. DEEP RESNET AND DEEP POLYNET

ResNet and PolyNet with one layer are derived from forward and backward Euler integration methods (see appendix for details). We can stack multiple layers to increase the stage and integration order, such as a ResNet with n layers (see Fig. 2):

$$h^1 = (I + \mathcal{G})(v) \tag{7}$$
$$h^2 = (I + \mathcal{G})(h^1) = (I + \mathcal{G})^2(v) \tag{8}$$
$$\vdots$$
$$h^n = (I + \mathcal{G})^n(v) \tag{9}$$

For a PolyNet with n layers (see Fig. 3):

$$h^1 = (I + \mathcal{G} + \mathcal{G}^2)(v) \tag{10}$$
$$h^2 = (I + \mathcal{G} + \mathcal{G}^2)(h^1) = (I + \mathcal{G} + \mathcal{G}^2)^2(v) \tag{11}$$
$$\vdots$$
$$h^n = (I + \mathcal{G} + \mathcal{G}^2)^n(v) \tag{12}$$

with h^n as the output of layer n (upper ^{n} denotes here the layer), and v the input. $\mathcal{G}$ is an inner non-linear transformation, which is considered as an FNN. Note that there is a modification of the original ResNet [35] and PolyNet [36], where in this work, $\mathcal{G}$ is the same FNN and they share weights and biases.

F. LINK TO DYNAMICAL SYSTEM IDENTIFICATION

To identify a discrete non-linear dynamical system, the prediction rule must first be determined. To predict $\bar{x}[k+1]$, the data used at the input of the neural network comprise $\bar{x}[k]$ and $\bar{u}[k]$. Then, the dynamical system retained is $\hat{x}[k+1] = g_{nn}(\bar{x}[k], \bar{u}[k])$, with $\bar{u}[k]$ as the true process input. With ResNet and PolyNet, it is not possible to replace v by $(\bar{x}[k],$

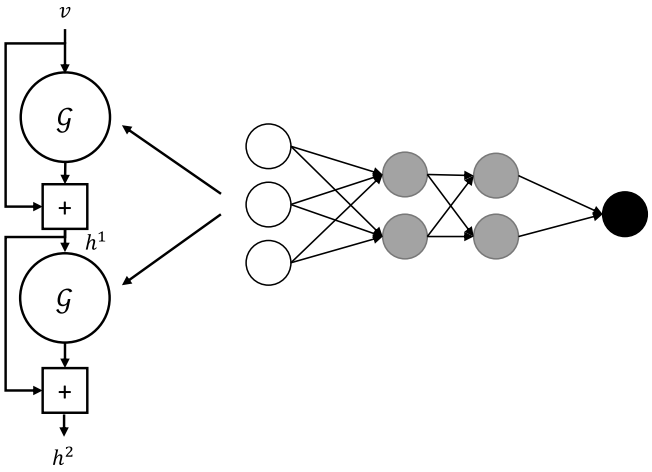

FIGURE 2. ResNet with two layers on the left. On the right, an illustration depicts an FNN which forms $\mathcal{G}$. White nodes are the inputs, gray nodes are the hidden units and the black node is the output (number of inputs, hidden layers and outputs are only for example).

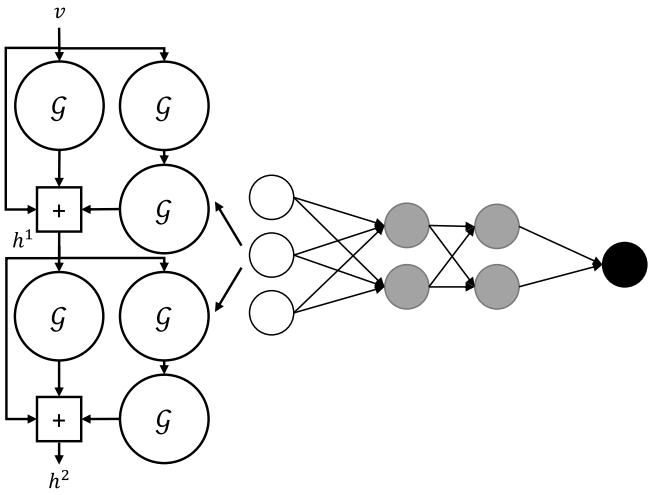

FIGURE 3. PolyNet with two layers on the left. On the right, an illustration depicts an FNN which forms $\mathcal{G}$. White nodes are the inputs, gray nodes are the hidden units and the black node is the output (number of inputs, hidden layers and outputs are only for example).

$\bar{u}[k]$) and h by ($\hat{x}[k+1]$) due to vector dimension issues with input/output vectors and with the number of neurons within the hidden layers. To remedy this problem, an input layer and output layer are added to the network (see Fig. 4). To adapt vector dimensions, neurons from the input and output layers have an identity activation function. The input layer and output layer are defined as:

$$h^{in} = W^{in}(\bar{x}[k], \bar{u}[k]) + b^{in} \tag{13}$$

$$\vdots$$

$$\hat{x}[k+1] = W^{out}(h^n) + b^{out} \tag{14}$$

with h^{in} as the output of the input layer such as $v := (\bar{x}[k], \bar{u}[k])$, W^{in}, b^{in} the weighting and bias of the input layer, W^{out}, b^{out} the weighting and bias of the output layer, $\bar{x}[k]$ the system state, $\bar{u}[k]$ the system input, $\hat{x}[k+1]$ the predicted state and h^n the layer n output.

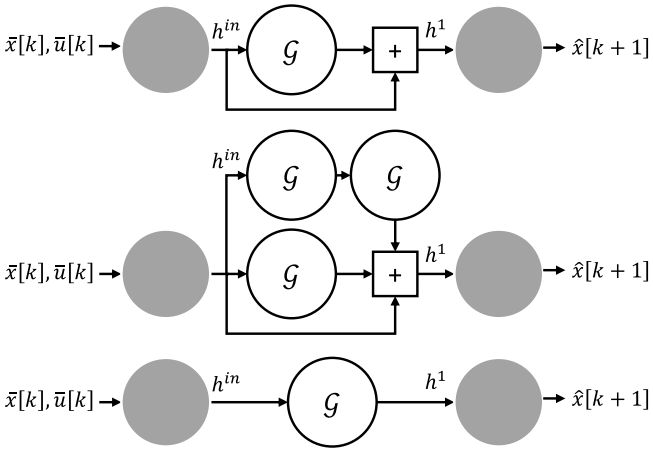

FIGURE 4. Illustration of closing the link with one layer of ResNet (top), one layer of PolyNet (middle) and FNN (bottom). Gray nodes form input and output layers of g_{nn}, which have an identity activation function.

IV. NEURAL TUNING

A. TRAINING

Neural network training aims to iteratively tune the neurons' weights and biases to minimise a fidelity measure [54]. Training is usually performed with the gradient-based back-propagation algorithm to adapt neurons' weights from the output layer to the input layer [59], [60].

The gradient-based back-propagation algorithm allows iteratively tunes weights and bias of the neurons, such as [61]:

$$W = W + \Delta W \tag{15}$$

$$b = b + \Delta b \tag{16}$$

with W the weighting vector, ΔW the increment vector, b the bias and Δb the increment vector. The increment vectors are:

$$\Delta W = \eta \frac{\partial L}{\partial W} \tag{17}$$

$$\Delta b = \eta \frac{\partial L}{\partial b} \tag{18}$$

with η the learning rate and L a loss function. Also, the loss function is defined in section IV-B. It has been observed that the learning rate is sensitive in regard to stability and training performance with respect to identification performance. Large value derives in lack of robustness in convergence and small value derives in lack of identification accuracy. In order to improve the training performance, one strategy is to gradually decrease the learning rate during training [61]. It can be, for instance, achieved by added an exponential learning rate schedule, such as [62]:

$$\eta_j = \eta_0 e^{-\lambda j} \tag{19}$$

with η_j as the updated learning rate, η_0 the initial learning rate, λ the decay parameter and j the iteration step. In

order to enhance neural network training performance, we selected stochastic gradient descent Adam optimizer [63] and derivative Radam [64], Nadam [65], Oadam [66]. Adam allows to fine tune the learning rate during training as it combines the adaptive gradient algorithm and root mean square propagation. The decay parameters of Adam were evaluated in [67], and results finding showed that choosing Adam parameters between 0.9 and 0.999 allows statistically a training improvement.

B. FIDELITY MEASURE

A loss function L provides a quantitative scoring that shows the degree of similarity between data and model outputs [68]. The fidelity measure considered in this work is the Mean Squared Error (MSE):

$$g_{mse} = \frac{1}{\text{card}(\mathcal{D})} \sum_{k \in \mathcal{D}} [\, \hat{x}[k+1] - \bar{x}[k+1] \,]^2 \,, \; (m^2) \quad (20)$$

with $\mathcal{D}$ as the number of samples, $\hat{x}[k+1]$ the neural network output and $\bar{x}[k+1]$ the target. Other loss functions exist but will not be considered: Mean Absolute Error (MAE), Root Mean Square Error (RMSE) or Mean Absolute Percentage Error (MAPE) [69].

C. HYPERPARAMETERS TUNING

The hyperparameters defining the neural network are the number of layers, number of neurons, type of activation function, batch size, epochs and the optimiser method for gradient-based backpropagation. These hyperparameters influence the performance and complexity of the model [70]. The hyperparameters are optimised according to:

$$\begin{aligned} \min_{H} \quad & g_{mse} && (21a) \\ \text{subject to} \quad & g_{nn}(H) && (21b) \\ & H \in \mathcal{H} && (21c) \end{aligned}$$

with g_{mse} as the cost function (see Eq. (20)), g_{nn} the neural network trained considered and $H \in \mathcal{H}$ the set of hyperparameters for the optimisation problem. The optimisation problem is non-convex and challenging since it is non-differentiable and involves constrained variables [70]. The metaheuristics algorithms may be considered, such as genetic algorithms or particle swarm optimisation [71]. The algorithm of the optimisation and interlinking with the backpropagation algorithm is shown in Fig. 5.

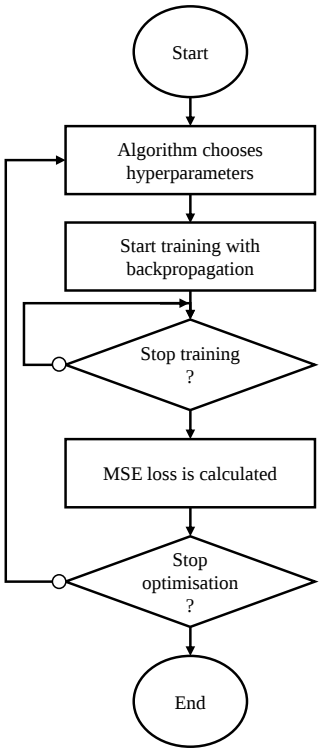

FIGURE 5. The algorithm of the optimisation of hyperparameters.

V. NEURAL NETWORK MODEL PREDICTIVE CONTROL

The neural networks presented in Section III are non-linear and involve non-convex optimisation when uses within MPC, and the resulting finding can be a local optimum [72]. A Neural Networks Model Predictive Control (NN-MPC) is considered with additional contractive state constraints, which is called a contractive MPC [73], [74]. The additional contractive constraints aim to ensure that the MPC is stabilised [73]:

$$\begin{aligned} \min_{\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{x}}} \quad & \sum_{i=0}^{N_h - 1} \tilde{x}_i^T Q \tilde{x}_i + \tilde{u}_i^T R \tilde{u}_i && (22a) \\ \text{subject to} \quad & \hat{x}_{i+1} = g_{nn}(\hat{x}_i, u_i) && (22b) \\ & \tilde{x}_i = \hat{x}_i - x_i^r && (22c) \\ & \tilde{u}_i = u_i - u_i^r && (22d) \\ & \hat{x}_0 = \bar{x}(t_k) && (22e) \\ & \hat{x} \in \mathcal{X} && (22f) \\ & u \in \mathcal{U} && (22g) \\ & \tilde{x}_{N_h} P \tilde{x}_{N_h} \leq \alpha \tilde{x}_0 P \tilde{x}_0 && (22h) \end{aligned}$$

Eq. (22a) is the quadratic cost function. Eq. (22b) is the state prediction according to the neural network presented in Section III. Eq. (22c) defines the state deviation from the state reference x^r. Eq. (22d) is the input deviation from the input reference u^r. Eq. (22e) is the state measurement. Eq. (22f) is the state constraint. Eq. (22g) is the input constraint. Eq. (22h) is the contractive constraint. $\tilde{\boldsymbol{u}}$ is the sequence of control inputs deviation $\tilde{\boldsymbol{u}} := (\tilde{u}_0, \ldots, \tilde{u}_{N_h - 1})$, and $\tilde{\boldsymbol{x}}$ is the sequence of predicted states deviation $\tilde{\boldsymbol{x}} := (\tilde{x}_0, ..., \tilde{x}_{N_h})$. At each iteration, the first sample of the optimal input computed is applied to the plant's actuators (Fig. 6). In addition, N_h denotes the time horizon considered for prediction. Q, R, P are the weighting matrices. $\alpha \in [0, 1[$ is the contractive parameter. P, Q and R are positive definite matrices, and ^{T}

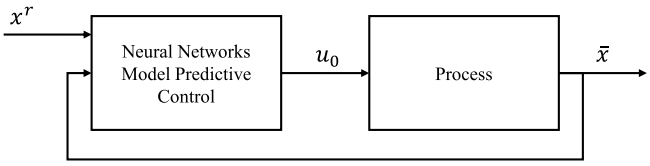

FIGURE 6. Control loop.

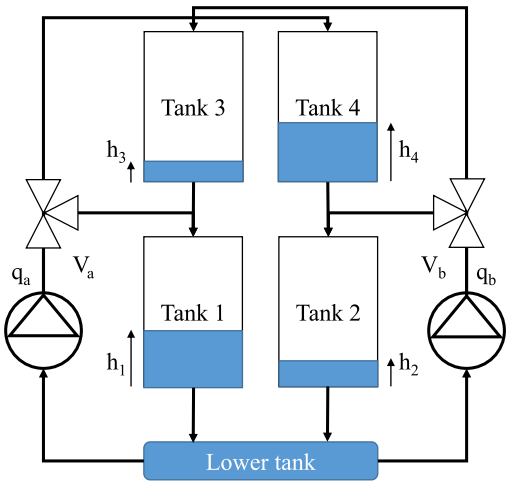

FIGURE 7. Four tank benchmark illustration.

denotes the matrix transpose.

VI. EXPERIMENTAL SETUP

A. QUADRUPLE TANK PROCESS DESCRIPTION

The considered benchmark has four tanks, two pumps and two three-way valves. The control has to manage a remain water level. Each tank has an orifice from which water leaks. Pump q_a fills tanks 1 and 4, and pump q_b fills tanks 2 and 3 (see Fig. 7).

B. MODELLING

QTP is modelled using Modelica [75] with Dymola software from Dassault Systèmes to support the simulation and generate data using fine-scale modelling rather than physical equations. Each component is modelled, including: water properties, pumps inertia, and pipe roughness, using the Modelica standard library [76] and building library [77]. The components are graphically assembled to build the QTP simulation model. An overflow outlet is added to the four tanks to consider the maximal water level and to avoid simulation breaking, whose process parameters are shown in Tab. 1. The Modelica four-tank program has 1,049 equations and the same number of unknown variables. These equations could not be used for the MPC controller due to some if-then-else conditions. A hybrid MPC controller is mandatory and leads to complex analyses, design and optimisation techniques [78].

TABLE 1. QTP parameters [41].

Parameters	Value	Units	Description
$h_{1_{max}}$	1.36	m	Maximum water level tank 1
$h_{2_{max}}$	1.36	m	Maximum water level tank 2
$h_{3_{max}}$	1.30	m	Maximum water level tank 3
$h_{4_{max}}$	1.30	m	Maximum water level tank 4
h_{min}	0.2	m	Minimum water level all tanks
$q_{a_{max}}$	3.26	m^3/h	Maximum water flow pump a
$q_{b_{max}}$	4.00	m^3/h	Maximum water flow pump b
q_{min}	0	m^3/h	Minimum water flow all pumps

TABLE 2. Parameters of the PRBS signals.

Parameters	Value	Units
Time period	500	s
Start time	500	s
End time	30	days
Amplitude	q_{min} or q_{max}	m^3/h

TABLE 3. Parameters of constant input signals.

Parameters	Value	Units
Time period	1200	s
Start time	30	days
End time	600	days
Amplitude	q_{min} to q_{max}	m^3/h

C. DATA GENERATION

The nature of the input signals is essential for dynamical system identification. The relation between the input signals and the outputs variations are used to describe the dynamical system. Pseudo-Random Binary Sequences (PRBS) as input signals are able to excite the system with a widespread frequency spectrum in order to acquire a unique set of parameters [17]. In this work, we performed the data acquisition of the QTP digitally using Dymola with the fine-scale dynamical system from Modelica program. The sampling period is equal to 5s and, Differential Algebraic System Solver (DASSL) is used during simulations to allow solving the Modelica model comprising mixed differential and algebraic equations [79]. The PRBS signals applied on pumps q_a and q_b are visible in Fig.8.a and outputs signals in Fig.8.b. Also, PRBS parameters are shown in Tab. 2. During simulations, it appears that all water levels are never in a steady state (Fig.8.b). In order to allow ANN to learn the QTP steady state, we chose to apply a second kind of input signals. They are piecewise constant signals with random values from minimum to maximum. Whose parameters are shown in Tab. 3. Corresponding inputs and outputs signals are shown in Fig.8.c and Fig.8.d. We chose to consider a large amount of data. As a result, 10 368 001 time steps were generated.

D. DATA SEPARATION

Simulation data are separated between train, validation and test data. Training data are used to optimise weights and biases; validation data are used for hyperparameters optimisation; and test data are used for neural network generalisation assessment. In classical approaches data are separated between train, validation and test data and are picked randomly

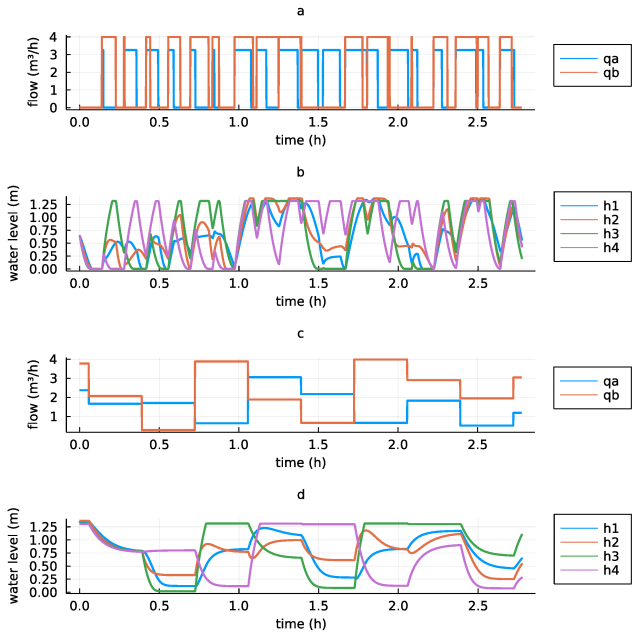

FIGURE 8. Piece of inputs/outputs generated. (a) Depicts PRBS inputs signals with pumps commands. (b) Depicts water level with PRBS inputs signals. (c) Depicts random constant inputs signals with pumps commands. (d) Depicts water level with random constant inputs signals.

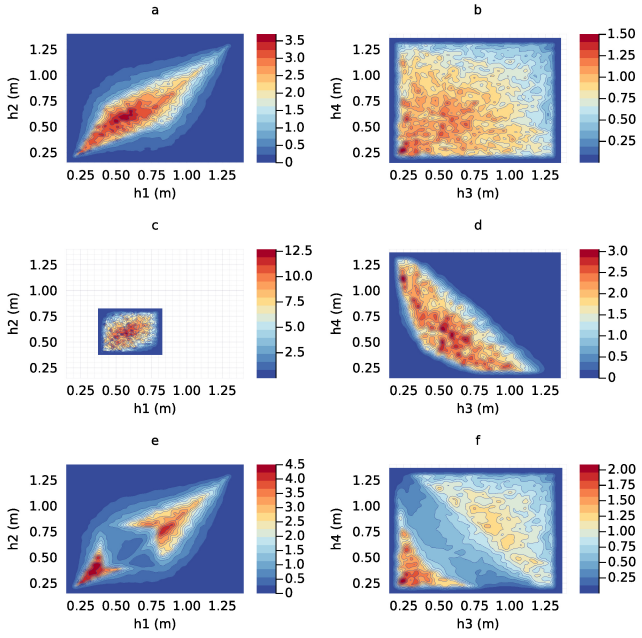

FIGURE 9. Kernel density estimation of data [80]. (a) Depicts kernel density estimation with h_1, h_2 water level and they are formed with the whole data. (b) Depicts kernel density estimation with h_3, h_4 water level and they are formed with the whole. (c) Depicts kernel density estimation with h_1, h_2 water level from 0.4m to 0.8m, and they are formed with the training and validation data. (d) Depicts kernel density estimation with h_3, h_4 water level with h_1, h_2 pairs from (a), and they are formed with the training and validation data. (e) Depicts kernel density estimation with h_1, h_2 outside water level from 0.4m to 0.8m, and they are formed with the test data. (f) Depicts kernel density estimation with h_3, h_4 water level with h_1, h_2 pairs from (e), and they are formed with the test data.

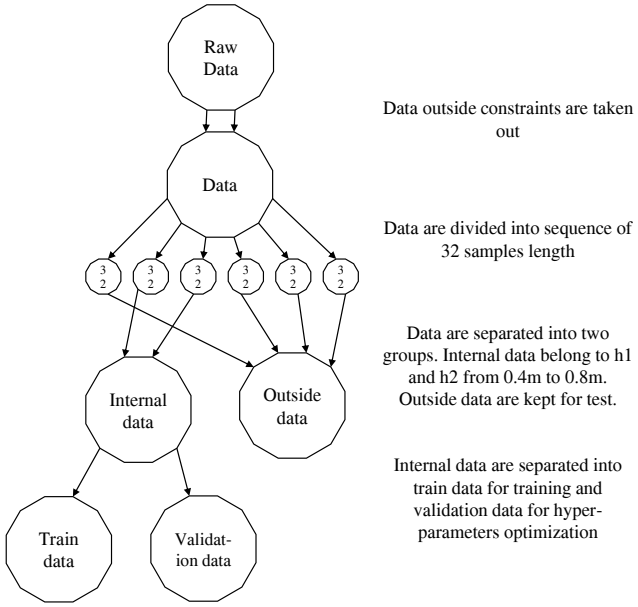

FIGURE 10. Procedure of data separation, from data generation to training, validation and test data.

from all subspace. To assess neural network architectures in this work, data separation is performed according to the tank levels' value ranges.

First, the data are processed to keep data within the water-level constraints (Fig.9.a and Fig.9.b); second, the data are divided into sequences of 32 samples length; third, the sequences are separated into two groups, where the first group data are picked when the sequenced water levels of tanks 1 and 2 (h_1, h_2) remained between 0.4 m to 0.8 m (Fig.9.c and Fig.9.d). The remaining data thus constituted the second group and to form the test data (e.g., water level lower than 0.4 m or greater than 0.8 m) (Fig.9.e and Fig.9.f). Finally, the first group is divided into two, with 5% for the validation data and the remaining for the training data. This data processing is shown in Fig. 10.

E. COMPUTING MACHINE IMPLEMENTATION

1) Neural network training

Neural networks are implemented on computing machines using the open-source Julia programming language [81] and Flux package [82]. This package allows easily implementing neurons, shortcut connections for ResNet, parallel connections for PolyNet, deep networks with multiple layers and all necessary training and fidelity measure computing. The hyperparameters are optimised using the BlackBoxOptim package which implements metaheuristics algorithms [83]. We chose the *separable NES* derived from Natural Evolution Strategies [84]. The hyperparameters considered in this work and their values are presented in Tab. 4. The hyperparameters optimisation is arbitrarily stopped when more than 1,000 network trainings are performed for each considered architecture (FNN, ResNet, PolyNet), resulting in 3,000 trained networks. The training is performed in parallel with 8 cores

TABLE 4. Hyperparameters range values.

Description	Range value
Activation function, Fig. 11	Softsign [88], Swish [89], Tanh [88], Sigmoid [88]
Neurons number per hidden layer	4 to 20
Hidden layer number	1 to 3
ResNet or PolyNet layer number	1 to 9
Epoch number	5 to 50
Batch size	32
Optimiser	adam [63], radam [64], nadam [65], oadam [66]
Learning rate	1.00×10^{-7} to 0.001
Momentum exponential decay	0.9 to 0.999
Momentum estimate	0.9 to 0.999

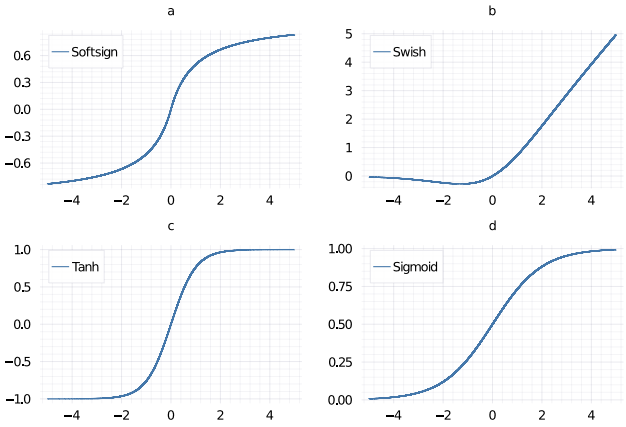

FIGURE 11. Activation function considered in this work. (a) Depicts Softsign activation function $\frac{x}{1+|x|}$. (b) Depicts Swish activation function $\frac{x}{1+e^{-x}}$ (c) Depicts Tanh activation function $tanh(x)$. (d) Depicts Sigmoid activation function $\frac{1}{1+e^{-x}}$.

on the Central Processing Unit (CPU), which leads to 125 steps for the metaheuristics algorithms. The running environment is composed of Windows 10 on Dell Workstation with Intel Xeon Gold 5122 CPU and 128 GB of random access memory.

2) Process control

Controller simulation is performed using Simulink. The Modelica model is first exported to Functional Mock-up Unit (FMU) for simulation. The controller interacts with the FMU according to Fig. 12 and is implemented through JuMP [85] using a multiple-shooting numerical method [72]. The neural network is defined with JuMP from the Flux object function as a *JuMP user-defined function*, which allows automatic differentiation to compute the derivative [87]. It is not necessary to rewrite the neural network within the optimisation modelling since Flux and JuMP are both in Julia's ecosystem. In addition, the chosen optimisation solver is Ipopt [86]. The set points used during simulations are shown in Tab. 5. The set-points are selected to evaluate control performance with different tank water levels. In addition, the set-points 1 and 3 are located inside the train data area and the set-points 2 and 4 are located outside the train data area. This choice will make possible to appreciate the extrapolation capability

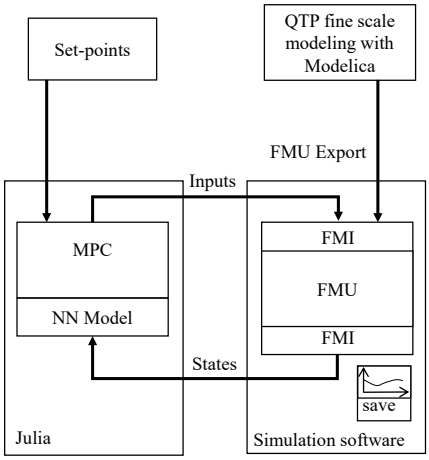

FIGURE 12. Simulator with the controller and the FMU.

of the ANNs. Set point 1 in Tab. 5 could be considered as the saddle point. It is implemented to initialize tanks water levels at the beginning of the simulation. In addition, the four tanks water level references are inside the train data area. Set point 2 increases difficulties. References get closer to the state constraints and they are outside the train data area. As a result, generalisation capabilities of the ANN are harnessed. Set point 3 added another difficulty. The four water levels are not equivalent. However, water levels remain within the train data area. Set point 4 is the most difficult to reach. It brings all the difficulties: references get closer to state constraints and the four tank water levels are not equivalent. In addition, generalisation capabilities of the ANN are harnessed as the set point 4 is located outside the train data area.

Several simulations are performed to analyse control performance regarding the implemented neural network. For each neural network architecture (FNN, ResNet, PolyNet), the 25 networks with the lowest MSE with test data are simulated, and for each simulation run, a comparative performance index is calculated to assess networks' influence on control. This index is defined as:

$$\mathrm{J} := \frac{1}{N_s} \sum_{k=2700}^{12000} e_x^T[k] Q e_x[k] + e_u^T[k] R e_u[k] \tag{23}$$

with N_s as the number of simulation samples, $e_x[k] = \bar{x}[k] - x^r[k]$ the state deviation from the reference, Q the weighting state matrix, $e_u[k] = \bar{u}[k] - u^r[k]$ the input deviation from the reference and R the weighting input matrix. In addition, the MPC parameters used during process control are shown in Tab. 6.

VII. RESULTS

A. SYSTEM IDENTIFICATION

The number of neural networks trained with the metaheuristic algorithm is 3,027 (1,009 for FNN, ResNet and Polynet). Neural networks with MSEs greater than 1 are removed, which results in 2,848 considered neural networks (918 for

TABLE 5. Set points used during simulations.

Reference	Time (s)	State (m)	Input (m^3/h)
Set-point 1	0 to 3000	$x_1 = 0.650$	$u_1 = 1.637$
		$x_2 = 0.650$	$u_2 = 1.988$
		$x_3 = 0.652$	
		$x_4 = 0.664$	
Set-point 2	3000 to 6000	$x_1 = 0.300$	$u_1 = 1.112$
		$x_2 = 0.300$	$u_2 = 1.351$
		$x_3 = 0.301$	
		$x_4 = 0.306$	
Set-point 3	6000 to 9000	$x_1 = 0.500$	$u_1 = 2.201$
		$x_2 = 0.750$	$u_2 = 1.361$
		$x_3 = 0.305$	
		$x_4 = 1.200$	
Set-point 4	9000 to 12000	$x_1 = 0.900$	$u_1 = 1.528$
		$x_2 = 0.750$	$u_2 = 2.539$
		$x_3 = 1.062$	
		$x_4 = 0.579$	

TABLE 6. Controller parameters.

Parameter	Range value
Q	diag(10 10 10 10)
R	diag(1 1)
N	20
α	0.90
P	diag(1 1 1 1)

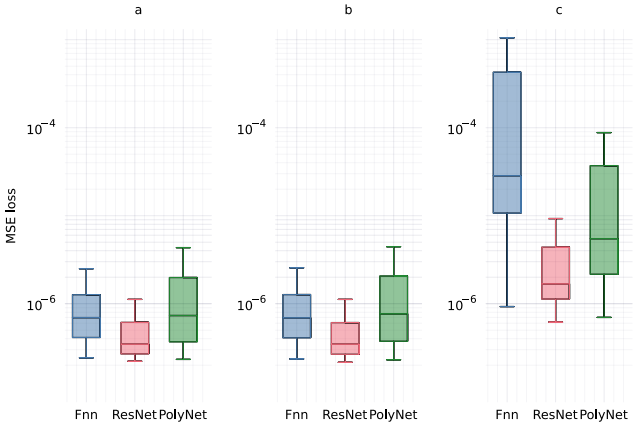

FIGURE 13. The boxplot loss with neural network architectures (FNN in blue, ResNet in red, PolyNet in green). (a) Depicts boxplot with training data. (b) Depicts boxplot with validation data. (c) Depicts boxplot with test data. Please note that outliers are not shown in order to clarify the figure.

FNN, 964 for ResNet and 966 for PolyNet), which are depicted hereafter.

Fig. 13 depicts the boxplot function loss of the neural networks after training and hyperparameters optimisation. The MSE with training data is visible in Fig. 13.a, MSE with validation data in Fig. 13.b and MSE with test data in Fig. 13.c. Fig. 13.a and Tab. 7 show that the 1^{st} quartile losses are reduced for neural networks related to numerical integration compared to the FNN. In addition, the lowest MSE is reached by ResNet for the median and 3^{rd} quartile, and the same is observable in Fig. 13.b and Tab. 8 for the MSE with the validation data. Note that all networks have a relatively equivalent performance in MSE with the

TABLE 7. MSE with train data.

Architecture	1^{st} quartile	Median	3^{rd} quartile
FNN	4.13×10^{-7}	6.96×10^{-7}	1.267×10^{-6}
ResNet	$\mathbf{2.69 \times 10^{-7}}$	$\mathbf{3.47 \times 10^{-7}}$	$\mathbf{6.14 \times 10^{-7}}$
PolyNet	3.69×10^{-7}	7.34×10^{-7}	1.99×10^{-6}

TABLE 8. MSE with validation data.

Architecture	1^{st} quartile	Median	3^{rd} quartile
FNN	4.10×10^{-7}	6.92×10^{-7}	1.27×10^{-6}
ResNet	$\mathbf{2.66 \times 10^{-7}}$	$\mathbf{3.47 \times 10^{-7}}$	$\mathbf{6.10 \times 10^{-7}}$
PolyNet	3.74×10^{-7}	7.68×10^{-7}	2.01×10^{-6}

training data and validation data, but the result is clearer when considering the generalisation properties, according to the results obtained with the test data in Fig. 13.c and Tab. 9. In this case, both neural networks related to the numerical integration method achieve a reduction of the median, 1^{st} and 3^{rd} quartile MSE compared to the FNN, and ResNet achieve the lowest MSE compared to the FNN and PolyNet.

The hyperparameters' influence over MSE with test data is shown in Fig. 14. The figure presents MSE with test data over hyperparameters optimisation with hidden layers, neurons, activation functions, optimisers and epochs for the FNN (first line), ResNet (second line) and PolyNet (third line).

First, boxplots of MSE over hidden layers for FNN, ResNet and PolyNet are depicted in Fig. 14.a, Fig. 14.f and Fig. 14.k. The results show that increasing the number of hidden layers increases the median MSE test loss for the FNN, with 1.34×10^{-5} with one hidden layer, 3.32×10^{-5} with two hidden layers and 3.86×10^{-5} with three hidden layers. Increasing the number of hidden layers reduces the median MSE with test data for ResNet and PolyNet, 2.59×10^{-6} and 7.09×10^{-5} with one hidden layer, 1.83×10^{-6} and 1.03×10^{-5} with two hidden layers, 1.51×10^{-6} and 4.16×10^{-6} with three hidden layers.

Second, we can analyse the impact of the number of neurons per hidden layer (see Fig. 14.b, Fig. 14.g, Fig. 14.l). The results show that the algorithm investigates 10 to 18 neurons for FNN, 6 to 15 neurons for ResNet and 4 to 15 neurons for PolyNet, while the neurons' values range from 4 to 20 neurons.

Third, we can audit the MSE with test data over the activation functions (Fig. 14.c, Fig. 14.h, Fig. 14.m). The swish activation function allows a reduction for FNN networks (Fig. 14.c); for instance, the median MSE is equal to 5.04×10^{-4} for sigmoid, 9.42×10^{-4} for softsign, 1.32×10^{-5} for swish and 1.77×10^{-4} for tanh. The ResNet median MSE with test data is equal to 1.34×10^{-6} for sigmoid, 1.73×10^{-6}

TABLE 9. MSE with test data.

Architecture	1^{st} quartile	Median	3^{rd} quartile
FNN	1.07×10^{-5}	2.85×10^{-5}	4.28×10^{-4}
ResNet	$\mathbf{1.13 \times 10^{-6}}$	$\mathbf{1.68 \times 10^{-6}}$	$\mathbf{4.39 \times 10^{-6}}$
PolyNet	2.16×10^{-6}	5.49×10^{-6}	3.71×10^{-5}

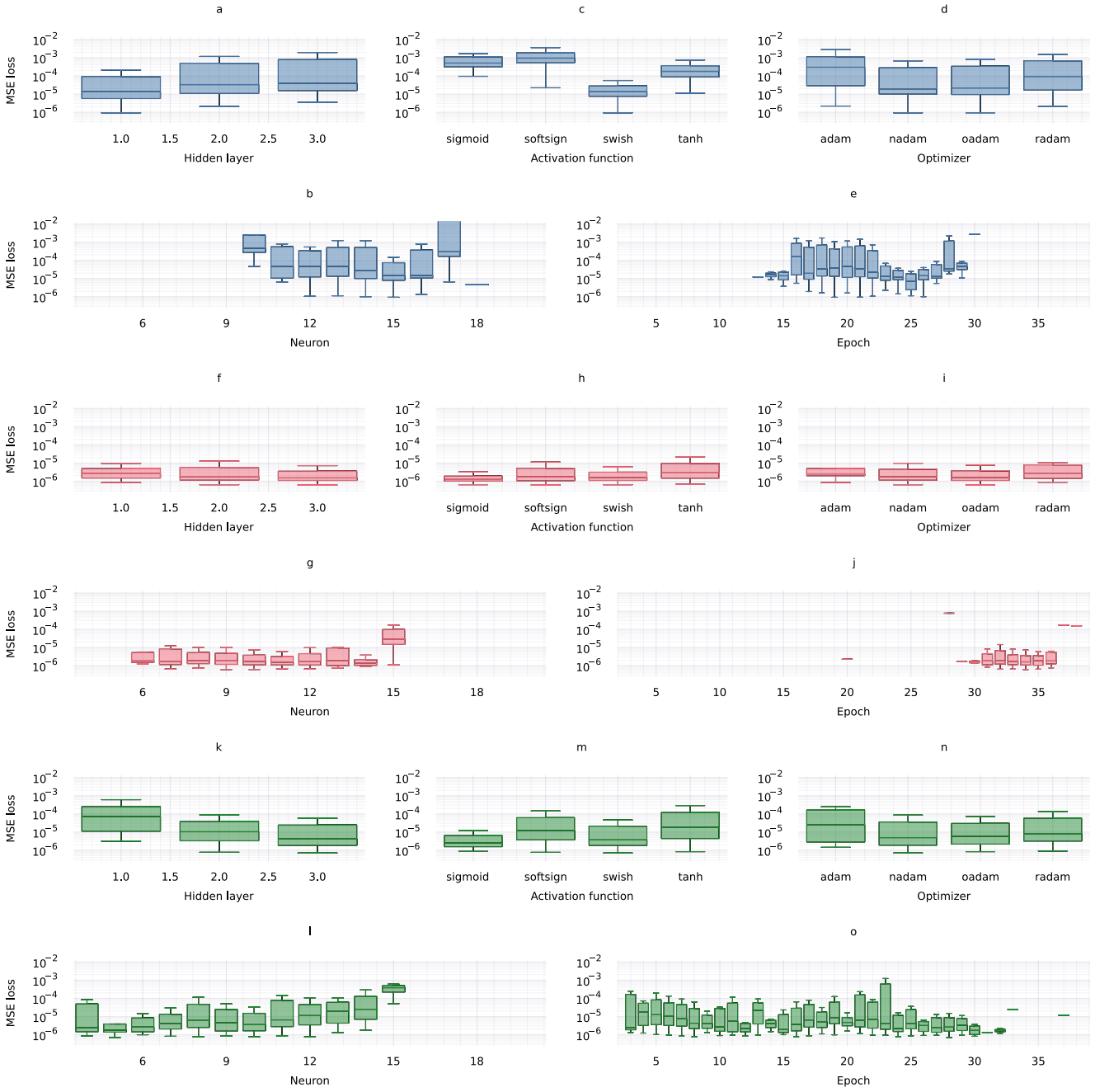

FIGURE 14. The boxplot loss with test data over hyperparameters optimisation (FNN in blue, ResNet in red, PolyNet in green). (a) (f) (k) Depict boxplot test loss over hidden layer. (c) (h) (m) Depict boxplot test loss over activation function. (d) (i) (n) Depict boxplot test loss over optimizer. (b) (g) (l) Depict boxplot test loss over neuron. (e) (j) (o) Depict boxplot test loss over epoch. Please note that outliers are not shown in order to clarify the figure.

for softsign, 1.60×10^{-6} for swish and 2.86×10^{-6} for tanh. The PolyNet median MSE with test data is equal to 2.54×10^{-6} for sigmoid, 1.19×10^{-5} for softsign, 3.67×10^{-6} for swish and 1.80×10^{-5} for tanh. The results show that the sigmoid allows a reduction of the median MSE for ResNet and PolyNet.

Fourth, we can explore MSE with test data over optimiser, adam, nadam, oadam and radam (Fig. 14.d, Fig. 14.i and Fig. 14.n). Note that nadam [65], oadam [66] and radam [64] are derived from adam [63]. The FNN median MSE with test data is equal to 2.98×10^{-4} for adam, 1.94×10^{-5} for nadam, 2.18×10^{-5} for oadam and 9.40×10^{-5} for radam. The ResNet median MSE with test data is equal to 2.53×10^{-6} for adam, 1.76×10^{-6} for nadam, 1.60×10^{-6} for oadam and 2.59×10^{-6} for radam. The PolyNet median MSE with test data is equal to 2.46×10^{-5} for adam, 4.93×10^{-6} for nadam, 5.81×10^{-6} for oadam and 7.70×10^{-6} for radam. Nadam and oadam allow a reduction of median test loss compared to

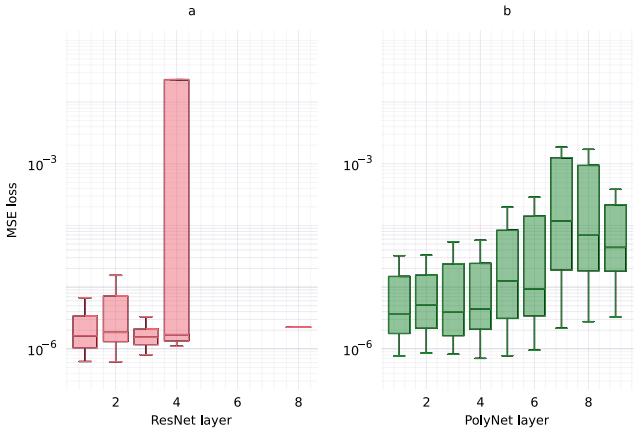

FIGURE 15. The boxplot loss with test data over layers (ResNet in red, PolyNet in green). (a) Depicts boxplot test loss with ResNet over layers. (b) Depicts boxplot test loss with PolyNet over layers. Please note that outliers are not shown in order to clarify the figure.

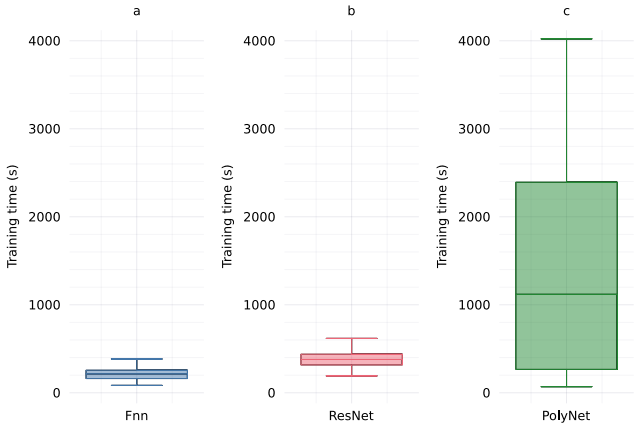

FIGURE 16. The boxplot training time with neural network architectures. (a) Depicts boxplot training time with FNN. (b) Depicts boxplot training time with ResNet. (c) Depicts boxplot training time with PolyNet. Please note that outliers are not shown in order to clarify the figure.

TABLE 10. Training time (s).

Architecture	1^{st} quartile	Median	3^{rd} quartile
FNN	**161.89**	**215.29**	**256.26**
ResNet	316.71	378.38	438.17
PolyNet	265.78	1118.32	2392.48

adam and radam.

Fifth, we can examine the MSE with test data over epochs (Fig. 14.e, Fig. 14.j and Fig. 14.o). The algorithm investigated 13 to 30 epochs for the FNN, 20 to 38 epochs for ResNet and 3 to 37 epochs for PolyNet. The epochs' allowed values range from 5 to 50, and thus the algorithm did not consider all possibilities. Fig. 14.e shows that increasing epochs from 15 to 25 allows a reduction of the MSE with test data, while it grows from 25 to 30. In Fig. 14.j with ResNet, the algorithm chose most epochs from 30 to 36, and the median MSE remains equivalent. Fig. 14.o shows that increasing the epoch allows a reduction of the median MSE with data until 30 epochs with PolyNet.

We can regard the ResNet and PolyNet number of layers, (see Fig. 15.a and Fig. 15.b). The algorithm investigated 1 to 4 and a jump to 8 layers for ResNet but from 1 to 9 for PolyNet. Fig. 15.a shows that the median MSE plateaus from 1 to 4 layers with 1.62×10^{-6} for one layer, 1.85×10^{-6} for two layers, 1.54×10^{-6} for three layers and 1.70×10^{-6} for four layers. Fig. 15.b shows that the median MSE with test data plateaus from 1 to 4 layers and increases from 5 to 9 layers with PolyNet. In addition, the smallest median MSE with test data is obtained with 3.62×10^{-6} and one layer.

Finally, Fig. 16 depicts boxplot training time with the FNN (Fig. 16.a), ResNet (Fig. 16.b) and PolyNet (Fig. 16.c), with values presented in Tab. 10. The median training time is equal to $215.29s$ for the FNN, $378.38s$ for ResNet and $1118.32s$ for PolyNet.

TABLE 11. Process control performance J.

	FNN-MPC	ResNet-MPC	PolyNet-MPC
Lowest	$\mathbf{2.567 \times 10^{-1}}$	2.576×10^{-1}	2.567×10^{-1}
1^{st} quartile	2.588×10^{-1}	2.582×10^{-1}	$\mathbf{2.580 \times 10^{-1}}$
Median	2.600×10^{-1}	$\mathbf{2.586 \times 10^{-1}}$	2.587×10^{-1}
3^{rd} quartile	2.627×10^{-1}	$\mathbf{2.592 \times 10^{-1}}$	2.612×10^{-1}
Highest	3.327×10^{-1}	$\mathbf{2.625 \times 10^{-1}}$	10.603

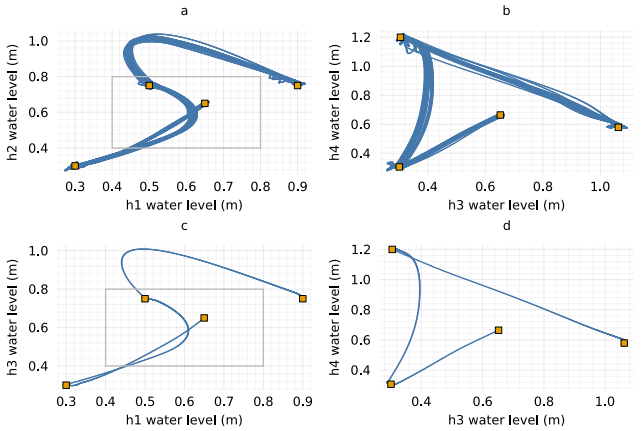

FIGURE 17. Depicts state trajectories with FNN-MPC, with set-points displayed as yellow squares. (a) h_1 over h_2 state trajectories for the 150 simulations. (b) h_3 over h_4 state trajectories for the 25 simulations. (c) h_1 over h_2 state trajectory for the lowest J. (d) h_3 over h_4 state trajectory for the lowest J. The gray square shows the area of the training data.

B. PROCESS CONTROL

Fig. 17 depicts simulation results with water level controlled by the FNN-MPC for the 25 simulations. Fig. 17.a and Fig. 17.b show that state trajectories are not the same according to the neural model. Fig. 17.c and Fig. 17.d show the lowest J with state trajectories from , $2700s$ to $12,000s$. The lowest J with FNN-MPC is equal to 2.567×10^{-1} (Tab. 11).

Fig. 18 depicts simulation results with water level controlled by the ResNet-MPC for the 25 simulations. Fig. 18.a and Fig. 18.b show that state trajectories are not the same

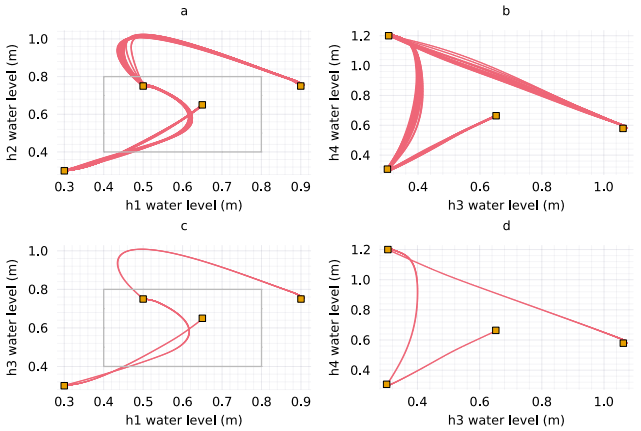

FIGURE 18. Depicts state trajectories with ResNet-MPC, with set-points displayed as yellow square. (a) h_1 over h_2 state trajectories for the 25 simulations. (b) h_3 over h_4 state trajectories for the 150 simulations. (c) h_1 over h_2 state trajectory for the lowest J. (d) h_3 over h_4 state trajectory for the lowest J. The gray square shows area of the training data.

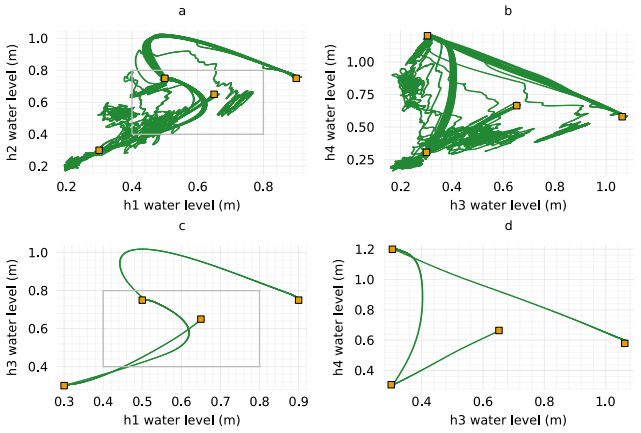

FIGURE 19. Depicts state trajectories with PolyNet-MPC, with set-points displayed as yellow square. (a) h_1 over h_2 state trajectories for the 25 simulations. (b) h_3 over h_4 state trajectories for the 25 simulations. (c) h_1 over h_2 state trajectory for the lowest J. (d) h_3 over h_4 state trajectory for the lowest J. The gray square shows area of the training data.

according to the neural model. Fig. 18.c and Fig. 18.d show the lowest J with state trajectories from $2,700s$ to $12,000s$. The lowest J with ResNet-MPC is equal to 2.576×10^{-1} (Tab. 11).

Fig. 19 depicts simulation results with water level controlled by the PolyNet-MPC for the 25 simulations. Fig. 19.a and Fig. 19.b show that state trajectories are not the same according to the neural model. Fig. 19.c and Fig. 19.d show the lowest J with state trajectories from $2,700s$ to $12,000s$. The lowest J with PolyNet-MPC is equal to 2.567×10^{-1} (Tab. 11).

Fig. 20 depicts boxplot J with controllers considered in this work, namely FNN-MPC (Fig. 20.a), ResNet-MPC (Fig. 20.b) and PolyNet-MPC (Fig. 20.c), with values shown in Tab. 11. The lowest J is observed with FNN-MPC, the lowest median for ResNet-MPC, the lowest 1^{st} quartile for PolyNet-MPC and the lowest 3^{rd} quartile for ResNet-MPC.

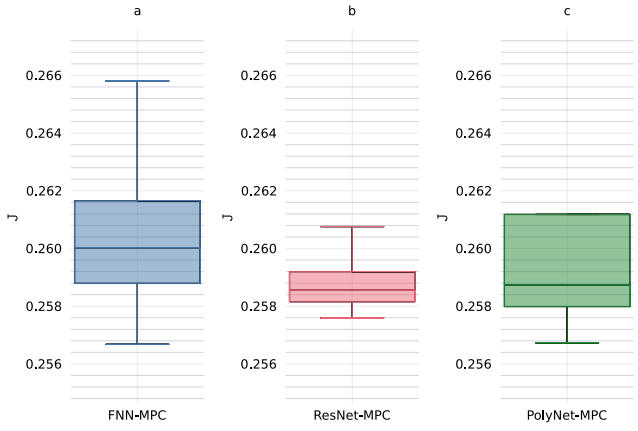

FIGURE 20. The boxplot J with neural networks based MPC without outliers, (a) FNN-MPC, (b) ResNet-MPC, (c) PolyNet-MPC. Please note that outliers are not shown in order to clarify the figure.

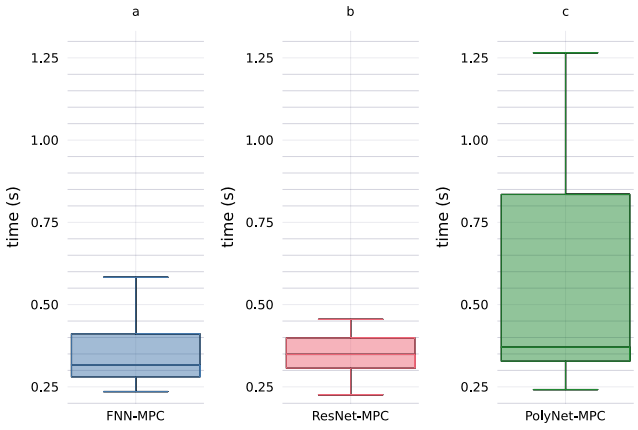

FIGURE 21. The boxplot mean computation time with neural networks based MPC without outliers, (a) FNN-MPC, (b) ResNet-MPC, (c) PolyNet-MPC. Please note that outliers are not shown in order to clarify the figure.

Fig. 21 depicts the boxplot mean computation time with regulators considered in this work, namely FNN-MPC (Fig. 21.a), ResNet-MPC (Fig. 21.b) and PolyNet-MPC (Fig. 21.c), with values presented in Tab. 12. The lowest mean computation time is observed with ResNet-MPC, the lowest median with FNN-MPC, the lowest 1^{st} quartile with FNN-MPC and the lowest 3^{rd} quartile with ResNet-MPC.

VIII. DISCUSSION

ResNet and PolyNet were originally presented for computer vision applications such as image classification, as they are

TABLE 12. Process control mean computation time (s).

	FNN-MPC	ResNet-MPC	PolyNet-MPC
Lowest	2.356×10^{-1}	$\mathbf{2.258 \times 10^{-1}}$	2.416×10^{-1}
1^{st} quartile	$\mathbf{2.801 \times 10^{-1}}$	3.067×10^{-1}	3.280×10^{-1}
Median	$\mathbf{3.164 \times 10^{-1}}$	3.487×10^{-1}	3.707×10^{-1}
3^{rd} quartile	4.111×10^{-1}	$\mathbf{3.988 \times 10^{-1}}$	8.350×10^{-1}
Highest	9.956×10^{-1}	$\mathbf{8.521 \times 10^{-1}}$	8.305

TABLE 13. Performance comparison results and compromises.

Evaluation criteria	FNN-MPC	ResNet-MPC	PolyNet-MPC
Identification error	++	+++	+
ANN generalisation	+	+++	++
ANN Training time	+++	++	+
J	++	+++	+
MPC computation time	++	+++	+

well suited for this task by using hundreds of layers for increasing accuracy, but they are only recently used for dynamical system identification. In this work, we observed that in the case of ResNet and PolyNet, the best performances were obtained with only a few layers, and that increasing the number of layers does not increase the dynamic identification accuracy of the system (range 1 to 9). ResNet seems to be a favourable option, while the performance of the FNN for dynamical system identification was shown to greatly depend on the chosen activation function, which is a point of weakness.

In this work, neural network identification for control purposes was further considered, specifically for MPC control. It was observed that the lowest identification error (MSE) did not systematically produce the lowest process control error (J).

The evaluation criteria are given in Tab.13 for each controller. They are rated from + to +++, with +++ better than +. ResNet-MPC produces the lowest identification error, the best performance control criterion J, the lowest computation time for the MPC. It has the best generalization ability compared to FNN-MPC and PolyNet-MPC. In addition, ResNet-MPC gives the most consistent results in control. Indeed, the results are good for a large choice of hyper-parameters, and homogeneous. The only drawback lies in the increase in training time compared to feedforward networks, the training was performed on CPUs only, and valuable reduction of training time could be achieved using specific hardware such as GPUs [90] or TPUs [91]. However, the performances of the PolyNet MPC are disappointing, in that we observe sometimes erratic simulation results (despite a rather favorable criterion). Hence the bad notation (+) next to Fnn-MPC and ResNet-MPC.

Finally, ResNet-MPC turns out to be the best solution, and a good candidate for systematic use in neural network based modeling and NN-MPC control of dynamical systems.

IX. CONCLUSION

ResNet and PolyNet neural networks are architectures that can be linked explicitly to Euler integration method. Thus, the initial motivation in this study was to evaluate their particular interest to deal with dynamical systems. Firstly, for non-linear identification, and secondly as a support to MPC implementation, using ad hoc numerical integration scheme for the prediction part. A contribution of this paper lies in the consideration of these architectures outside their usual context of use, e.g. as deep networks for image recognition.

To assess the relevance of these networks for regression and control, the following choices were made. The first choice consisted in considering a representative multivariable system with coupled dynamics, illustrating certain classical characteristics of industrial processes. The second consisted in proposing a methodology to ensure a fair comparison of ResNet and PolyNet with the classical FNN. Common evaluation criteria have been defined. A systematic investigation of hyperparameters via metaheuristic optimization has been proposed. Finally, the data have been segmented in an original way to highlight the generalization capabilities.

The results of this work confirm, first of all, the particular interest of the two architectures ResNet and PolyNet studied. ResNet and PolyNet achieve equivalent performance on learned data. It is even observed, which was unexpected, that the generalization capacities of the ResNet surpass those of the PolyNet, in spite of the more complex architecture of the latter. Beyond that, we analyzed the quality of MPC water-level control obtained via MPC, considering in turn FNN, ResNet and PolyNet based prediction models. ResNet once again appears to be superior, both in terms of the homogeneity of the results obtained and the average computation time of the resulting MPC, which is lower than with PolyNet.

The perspectives of this work are multiple. To be concise, the work now consists of implementing the proposed methodology within the framework of the control of an agricultural greenhouse. At the same time, we are working more fundamentally on the problem of physics informed learning, in order to enrich the potential of the methodology in the case of a small volume of experimental data.

APPENDIX

For an ordinary DE $\dot{y}(t) = g(y(t))$, $y(0) = y_0$, the forward Euler integration method is defined as [9]:

$$y[k+1] = y[k] + T_s g(y[k]) \tag{24}$$

with $y[k+1]$ as the solution at $k+1$ and T_s the sample time. The link with ResNet is observable with Eq. (5). In addition, the backward Euler integration method is [9]:

$$y[k+1] = y[k] + T_s g(y[k+1]) \tag{25}$$

$$y[k+1] - T_s g(y[k+1]) = y[k] \tag{26}$$

$$y[k+1] = (I - T_s g)^{-1} y[k] \tag{27}$$

with $y[k+1]$ as the solution at $k+1$ and T_s the sample time. Then, the link with PolyNet and Eq. (6) is provided in [33] with: $\frac{1}{1-x} \approx \sum_{n=0}^{\infty} x^n$, such as $(I - T_s g)^{-1} \approx I + T_s g + (T_s g)^2 + \ldots + (T_s g)^n + \ldots$.

ACKNOWLEDGMENT

The authors thank the Julia community for online comments about the programming language.

REFERENCES

[1] D. Q. Mayne, "Model predictive control: recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, Dec. 2014, doi: 10.1016/j.automatica.2014.10.128

[2] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, July 2003, doi: 10.1016/S0967-0661(02)00186-7

[3] J. B. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Syst. Mag.*, vol. 20, no. 3, pp. 38–52, June 2000, doi: 10.1109/37.845037.

[4] K. R. Muske and J. B. Rawlings, "Model predictive control with linear models," *AIChE J.*, vol. 39, no. 2, pp. 262–287, Feb. 1993, doi: 10.1002/aic.690390208.

[5] L. Grüne and J. Pannek, "Nonlinear Model Predictive Control," in *Nonlinear Model Predictive Control: Theory and Algorithms*. Cham, Switzerland: Springer, 2017, ch. 3, pp. 43–66, doi: 10.1007/978-0-85729-501-9_3

[6] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Comput. Chem. Eng.*, vol. 23, no. 4-5, pp. 667–682, May 1990, doi: 10.1016/S0098-1354(98)00301-9.

[7] A. Ü. Keskin, "Basic concepts," in *Ordinary Differential Equations for Engineers*. Cham, Switzerland: Springer, 2019, ch. 1, pp 1–7, doi: 10.1007/978-3-319-95243-7_1.

[8] G. Sánchez, M. Murillo, L. Genzelis, N. Deniz and L. Giovanini, "MPC for nonlinear systems: A comparative review of discretization methods," *2017 XVII Workshop on Information Processing and Control (RPIC)*, 2017, pp. 1-6, doi: 10.23919/RPIC.2017.8214333.

[9] E. Hairer, G. Wanner and S. P. Nørsett, "Classical mathematical theory," in *Solving Ordinary Differential Equations I*, Berlin, Heidelberg, Germany: Springer, 2008, ch. 1, pp. 1–128, doi: 10.1007/978-3-540-78862-1_1.

[10] L. F. Shampine, "Solving ODEs and DDEs with residual control," *Appl. Numer. Math.*, vol. 52, no. 1, pp. 113–127, Jan. 2005, doi: 10.1016/j.apnum.2004.07.003.

[11] P. Bogacki and L. F. Shampine, "A 3 (2) pair of Runge-Kutta formulas," *Appl. Math. Lett.*, vol. 2, no. 4, pp. 321–325, 1989, doi: 10.1016/0893-9659(89)90079-7.

[12] E. Hairer, G. Wanner and S. P. Nørsett, "Multistep methods and general linear methods," in *Solving Ordinary Differential Equations I*, Berlin, Heidelberg, Germany: Springer, 2008, ch. 3, pp. 355–474, doi: 10.1007/978-3-540-78862-1_3.

[13] E. T. Maddalena, Y. Lian and C. N. Jones, "Data-driven methods for building control — A review and promising future directions," *Control Eng. Pract.*, vol. 95, art. no. 104211, Feb. 2020, doi: 10.1016/j.conengprac.2019.104211.

[14] C. Tsay and M. Baldea: "110th anniversary: using data to bridge the time and length scales of process systems," *Ind. Eng. Chem. Res*, vol. 58, no. 36, pp. 16696–16708, 2019, doi: 10.1021/acs.iecr.9b02282.

[15] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[16] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[17] I. D. Landau, G. Zito, "System identification: the bases," in *Digital Control Systems. Design, Identification and Implementation*. London, UK, Springer, 2006, ch. 5, pp. 201–236, doi: 10.1007/978-1-84628-056-6_5

[18] E. Terzi, T. Bonetti, D. Saccani, M. Farina, L. Fagiano and R. Scattolini, "Learning-based predictive control of the cooling system of a large business centre," *Control Eng. Pract.*, vol. 97, art. no. 104348, Apr. 2020, doi: 10.1016/j.conengprac.2020.104348.

[19] T. Fukuda and T. Shibata, "Theory and applications of neural networks for industrial control systems," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 472-489, Dec. 1992, doi: 10.1109/41.170966.

[20] K. J. Hunt, D. Sbarbaro, R. Żbikowski and P. J. Gawthrop, "Neural networks for control systems—a survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, Nov. 1992, doi: 10.1016/0005-1098(92)90053-I.

[21] J. G. Kuschewski, S. Hui and S. H. Zak, "Application of feedforward neural networks to dynamical system identification and control," *IEEE Trans. Control Syst. Technol.*, vol. 1, no. 1, pp. 37–49, March 1993, doi: 10.1109/87.221350.

[22] A. Ouaret, H. Lehouche, B. Mendil and H. Guéguen, "Supervisory control of building heating system with insulation changes using three architectures of neural networks," *J. Frankl. Inst.*, vol. 357, no. 18, pp. 13362–13385, Dec. 2020, doi: 10.1016/j.jfranklin.2020.09.027.

[23] P. Loiseau, C. N. E. Boultifat, P. Chevrel, F. Claveau, S. Espié and F. Mars, "Rider model identification: neural networks and quasi-LPV models," *IET Intell. Transp. Syst.*, vol. 14, no. 10, pp. 1259–1264, Oct. 2020, doi: 10.1049/iet-its.2020.0088.

[24] A. Yazdizadeh and K. Khorasani, "Adaptive time delay neural network structures for nonlinear system identification," *Neurocomputing*, vol. 47, no. 1-4, pp. 207–240, Aug. 2002, doi: 10.1016/S0925-2312(01)00589-6.

[25] A. Delgado, C. Kambhampati and K. Warwick, "Dynamic recurrent neural network for system identification and control," *Inst. Elect. Eng. Proc. Contr. Theory Applicat*, vol. 142, no. 4, pp. 307–314, July 1995, doi: 10.1049/ip-cta:19951873.

[26] Z. Wu and P. D. Christofides, "Economic machine-learning-based predictive control of nonlinear systems," *Mathematics*, vol .7, no. 6, art. no. 494, June 2019, doi: 10.3390/math7060494.

[27] W. C. Wong, E. Chee, J. Li and X. Wang, "Recurrent neural network-based model predictive control for continuous pharmaceutical manufacturing," *Mathematics*, vol. 6, no. 11, art. no. 242, Nov. 2018, doi: 10.3390/math6110242.

[28] Y. Wang, "A new concept using LSTM Neural Networks for dynamic system identification," *2017 American Control Conference (ACC)*, Seattle, WA, USA, 2017, pp. 5324–5329, doi: 10.23919/ACC.2017.7963782.

[29] J. Gonzalez and W. Yu, "Non-linear system modeling using LSTM neural networks," *IFAC Conference on Modelling, Identification and Control of Nonlinear Systems (MICNON)*, Guadalajara, Mexico, 2018, pp. 485–489, doi: 10.1016/j.ifacol.2018.07.326.

[30] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangöz and C. Jones, "Recurrent Neural Network based MPC for Process Industries," *2019 18th European Control Conference (ECC)*, Naples, Italy, 2019, pp. 1005–1010, doi: 10.23919/ECC.2019.8795809.

[31] A. Rehmer and A. Kroll, "On Using Gated Recurrent Units for Nonlinear System Identification," *2019 18th European Control Conference (ECC)*, 2019, pp. 2504-2509, doi: 10.23919/ECC.2019.8795631.

[32] E. Weinan, "A proposal on machine learning via dynamical systems," *Commun. Math. Stat.*vol. 5, no. 1, pp. 1–11, Mar. 2017, doi: 10.1007/s40304-017-0103-z.

[33] Y. Lu, A. Zhong, Q. Li and B. Dong, "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations," *International Conference on Machine Learning (PMLR)*, 2018, pp. 3276–3285.

[34] S. Ouala, A. Pascual and R. Fablet, "Residual Integration Neural Network," *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3622–3626, doi: 10.1109/ICASSP.2019.8683447.

[35] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[36] X. Zhang, Z. Li, C. C. Loy and D. Lin, "PolyNet: A Pursuit of Structural Diversity in Very Deep Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3900-3908, doi: 10.1109/CVPR.2017.415.

[37] Y.-J. Wang and C.-T. Lin, "Runge-Kutta neural network for identification of dynamical systems in high accuracy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 9, no. 2, pp. 294-307, Mar. 1998, doi: 10.1109/72.661124.

[38] M. Zhu, B. Chang and C. Fu, "Convolutional Neural Networks combined with Runge-Kutta Methods," *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.

[39] D. Bensoussan, *Commande moderne: approche par modèles continus et discrets*. Montreal, QC, Canada: Presses Internationales Polytechnique, 2008.

[40] K. H. Johansson, "The quadruple-tank process: a multivariable laboratory process with an adjustable zero," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 456-465, May 2000, doi: 10.1109/87.845876.

[41] I. Alvarado, D. Limon, D. M. De La Peña, J. M. Maestre, M. A. Ridao, H. Scheu, W. Marquardt, R. R. Negenborn, B. De Schutter, F. Valencia and J. Espinosa, "A comparative analysis of distributed MPC techniques applied to the HD-MPC four-tank benchmark," *J. Process Control*, vol. 21, no. 5, pp. 800–815, June 2011, doi: 10.1016/j.jprocont.2011.03.003.

[42] A. D'jorge, A. Anderson, A. H. González and A. Ferramosca, "A robust economic MPC for changing economic criterion," *International Journal of Robust and Nonlinear Control*, vol. 28, no. 15, pp. 4404–4423, Oct. 2018, doi: 10.1002/rnc.4243.

[43] S. B. Prusty, S. Seshagiri, U. C. Pati and K. K. Mahapatra, "Sliding mode control of coupled tank systems using conditional integrators," *IEEE/CAA J. Autom. Sin.*, vol. 7, no. 1, pp. 118–125, Jan. 2020, doi: 10.1109/JAS.2019.1911831.

[44] J. Berner, K. Soltesz, T. Hägglund and K. J. Åström, "An experimental comparison of PID autotuners," *Control Eng. Pract.*, vol. 73, pp. 124–133, Apr. 2018, doi: 10.1016/j.conengprac.2018.01.006.

[45] G. A. Hicks and W. H. Ray, "Approximation methods for optimal control synthesis," *Can. J. Chem. Eng.*, vol. 49, no. 4, pp. 522–528, Aug. 1971, doi: 10.1002/cjce.5450490416.

[46] M. Ellis and P. D. Christofides, "Integrating dynamic economic optimization and model predictive control for optimal operation of nonlinear process systems," *Control Eng. Pract.*, vol. 22, pp. 242–251, Jan. 2014, doi: 10.1016/j.conengprac.2013.02.016.

[47] D. Q. Mayne, E. C. Kerrigan, E. J. Van Wyk and P. Falugi, "Tube-based robust nonlinear model predictive control," *Int. J. Robust Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, July 2011, doi: 10.1002/rnc.1758.

[48] M. Hertneck, J. Köhler, S. Trimpe and F. Allgöwer, "Learning an approximate model predictive controller with guarantees," *IEEE Contr. Syst. Lett*, vol. 2, no. 3, pp. 543–548, July 2018, doi: 10.1109/LCSYS.2018.2843682.

[49] N. Magdelaine, "Diabète de type 1: du modèle... à la boucle fermée," Ph.D. dissertation, Centrale Nantes, Nantes, 2017, hal: tel-02889375.

[50] D. A. Copp, R. Gondhalekar and J. P. Hespanha, "Simultaneous model predictive control and moving horizon estimation for blood glucose regulation in type 1 diabetes," *Optim. Control Appl. Methods*, vol. 39, no. 2, pp. 904–918, Mar./Apr. 2018, doi: 10.1002/oca.2388.

[51] A. Grancharova and I. Valkova, "Contractive Model Predictive Control for Insulin Delivery for Type 1 Diabetics," *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*, 2019, pp. 1-6, doi: 10.1109/INISTA.2019.8778202.

[52] R. Chai, A. Savvaris, A. Tsourdos, S. Chai and Y. Xia, "Optimal tracking guidance for aeroassisted spacecraft reconnaissance mission based on receding horizon control," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 54, no. 4, pp. 1575–1588, Aug. 2018, doi: 10.1109/TAES.2018.2798219.

[53] R. Chai, A. Tsourdos, H. Gao, Y. Xia and S. Chai, "Dual-loop tube-based robust model predictive attitude tracking control for spacecraft with system constraints and additive disturbances," *IEEE Trans. Ind. Electron*, vol. 69, no. 4, pp. 4022–4033, Apr. 2022, doi: 10.1109/TIE.2021.3076729.

[54] B. S. Sousa, F. V. Silva and A. M. F. Fileti, "Level control of coupled tank system based on neural network techniques," *Chem. Prod. Process Model.*, vol. 15, no. 3, Sept. 2020, doi: 10.1515/cppm-2019-0086.

[55] J. Xu, C. Li, X. He and T. Huang, "Recurrent neural network for solving model predictive control problem in application of four-tank benchmark," *Neurocomputing*, vol. 190, pp. 172–178, May 2016, doi: 10.1016/j.neucom.2016.01.020.

[56] F. Verhulst, "Mathematics is the art of giving the same name to different things: An interview with Henri Poincaré," vol. 5/13, no. 3, pp. 154–158, Sept. 2012.

[57] X. S. Yang, *Nature-inspired metaheuristic algorithms*. Frome, United Kingdom: Luniver Press, 2010.

[58] K. K. K. Kim, E. R. Patron and R. D. Braatz, "Standard representation and unified stability analysis for dynamic artificial neural network models," *Neural Netw.*, vol. 98, pp. 251–262, Feb. 2018, doi: 10.1016/j.neunet.2017.11.014.

[59] Y. Lecun, "Une Procedure d'apprentissage pour reseau a seuil assymetrique," *Cognitiva 85*, June 1985.

[60] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[61] R. Chai, A. Tsourdos, A. Savvaris, S. Chai, Y. Xia and C. L. P. Chen, "Design and implementation of deep neural network-based control for automatic parking maneuver process," *EEE Trans. Neural Netw. Learn. Syst.* , 2020, doi: 10.1109/TNNLS.2020.3042120.

[62] J. Park, D. Yi, S. Ji, "A novel learning rate schedule in optimization for neural networks and it's convergence," *Symmetry*, vol. 12, no. 4, pp. 660, 2020, doi: 10.3390/sym12040660.

[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[64] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, 2020.

[65] T. Dozat, "Incorporating nesterov momentum into adam," *Workshop track - International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[66] C. Daskalakis, A. Ilyas, V. Syrgkanis and H. Zeng, "Training GANs with Optimism," *International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, 2018.

[67] P. C. Blaud, P. Chevrel, F. Claveau, P. Haurant, A. Mouraud, "From multiphysics models to neural network for predictive control synthesis," *Optim. Control. Appl. Meth.*, 2021, doi:10.1002/oca.2845.

[68] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98-117, Jan. 2009, doi: 10.1109/MSP.2008.930649.

[69] G. Li and J. Shi, "On comparing three artificial neural networks for wind speed forecasting," *Appl. Energy*, vol. 87, no. 7, pp. 2313–2320, July 2010, doi: 10.1016/j.apenergy.2009.12.013.

[70] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[71] M. Abdel-Basset, L. Abdel-Fatah and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Intelligent Data-Centric Systems, Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*. Academic Press, 2018, ch. 10, pp. 185–231, doi: 10.1016/B978-0-12-813314-9.00010-4.

[72] L .Grüne and J. Pannek, "Numerical optimal control of nonlinear systems," in *Nonlinear Model Predictive Control: Theory and Algorithms*. Cham, Switzerland, Springer, 2017, ch. 12, pp. 367–434, doi: 10.1007/978-3-319-46024-6_12.

[73] S. L. de Oliveira Kothare and M. Morari, "Contractive model predictive control for constrained nonlinear systems," *IEEE Trans. Autom. Control*, vol. 45, no. 6, pp. 1053-1071, June 2000, doi: 10.1109/9.863592.

[74] D. Q. Mayne, J. B. Rawlings, C. V. Rao and P. O. M. Scokaertet, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000, doi: 10.1016/S0005-1098(99)00214-9.

[75] S. E. Mattsson, H. Elmqvist and M. Otter, "Physical system modeling with Modelica," *Control Eng. Pract.*, vol. 6, no. 4, pp. 501–510, Apr. 1998, doi: 10.1016/S0967-0661(98)00047-1.

[76] The Modelica Association, "Modelica standard library," GitHub repository, 2020. Available: https://github.com/modelica/ModelicaStandardLibrary.

[77] M. Wetter, "Modelica-based modelling and simulation to support research and development in building energy and control systems," *J. Build. Perform. Simul.* vol. 2, no. 2, pp. 143–161, May 2009, doi: 10.1080/19401490902818259.

[78] M. Lazar, "Model predictive control of hybrid systems: stability and robustness," Technische Universiteit Eindhoven, Eindhoven, 2006, doi: 10.6100/IR612103.

[79] L. Liu, F. Felgner and G. Frey, "Comparison of 4 numerical solvers for stiff and hybrid systems simulation," *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*, 2010, pp. 1–8, doi: 10.1109/ETFA.2010.5641330.

[80] A. Bernacchia and S. Pigolotti, "Self-consistent method for density estimation, " *J. R. Stat. Soc. Series B Stat. Methodol.*, vol. 73, pp. 407–422, June 2011, doi: 10.1111/j.1467-9868.2011.00772.x.

[81] J. Bezanson, A. Edelman, S. Karpinski and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Rev.*, vol. 59, no. 1, pp. 65–98, Feb. 2017, doi: 10.1137/141000671.

[82] M. Innes, "Flux: elegant machine learning with Julia," *J. Open Res. Softw.*, vol. 3, no. 25, May 2018, doi: 10.21105/joss.00602.

[83] R. Feldt and A. Stukalov, "BlackBoxOptim.jl," GitHub repository, 2018. Available: https://github.com/robertfeldt/BlackBoxOptim.jl.

[84] N. Hansen, D. V. Arnold and A. Auger, "Evolution Strategies," in *Handbook of Computational Intelligence*. Berlin, Germany: Springer, Berlin, 2015, ch. 44, pp 871–898, doi: 10.1007/978-3-662-43505-2_44

[85] I. Dunning, J. Huchette and M. Lubin, "JuMP: A modeling language for mathematical optimization," *SIAM Rev.*, vol. 59, no. 2, pp. 295–320, May 2017, doi: 10.1137/15M1020575.

[86] A. Wächter and T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006, doi: 10.1007/s10107-004-0559-y.

[87] J. Revels, M. Lubin and T. Papamarkou, "Forward-mode automatic differentiation in Julia," arXiv:1607.07892, 2016. Available: https://arxiv.org/abs/1607.07892.

[88] D. Smith and H. LeBlanc, "Variable activation functions and spawning in neuroevolution," *ASEE North Central Section Conference*, Akron, Ohio, USA, 2018.

[89] P. Ramachandran, B. Zoph and Q. V. Le, "Searching for activation functions," *International Conference on Learning Representations (ICLR)*, Vancouver, BC, Canada, 2018.

[90] T. Besard, C. Foket and B. De Sutter, "Effective extensible programming: unleashing Julia on GPUs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 4, pp. 827–841, Apr. 2019, doi: 10.1109/TPDS.2018.2872064.

[91] N. Jouppi, C. Young, N. Patil and D. Patterson, "Motivation for and Evaluation of the First Tensor Processing Unit," *IEEE Micro*, vol. 38, no. 3, pp. 10–19, May/June 2018, doi: 10.1109/MM.2018.032271057.

PIERRE CLÉMENT BLAUD received the Master degree in control engineering from the Ecole Centrale de Nantes, Nantes, France, M.Ing. in electrical engineering from the ETS, Montréal, QC, Canada and the engineering diploma from the UTBM, Belfort, France. He is currently pursuing the Ph.D. degree with the CEA, Bouguenais, France, and the IMT Atlantique, Nantes, France. His research interests include neural networks and control of multi-carrier energy systems.

PHILIPPE CHEVREL (Member, IEEE) received the Ph.D. degree from the University of Paris XI, Orsay, France. He is currently a Professor with the IMT Atlantique, Nantes, France. He is the Head of the Control Team, Laboratory of Digital Sciences of Nantes (LS2N), and a member of two research groups of the CNRS such as MOSAR (multivariable robust control) and GTAA (automotive control). His research interests include robust and resilient control theory, active control, and multiobjective and structured estimation and control. He is also a member of the EEA and the IEEE Control Society.

FABIEN CLAVEAU received the Ph.D. degree in automatic control from the Ecole Centrale de Nantes, University of Nantes, Nantes, France. Since 2005, he has been an Associate Professor with the IMT Atlantique, Nantes, and is also a member of the Control Team, Laboratory of Digital Sciences of Nantes (LS2N), Nantes. His research interests include robust control, and decentralized and distributed control.

PIERRICK HAURANT received his PhD degree in energy at university of Corsica (France) in 2012. He is associate professor at IMT Atlantique since 2016 in the department energy systems and environment and also a member of the GEPEA CNRS (French National Centre for Research) Joint Research Lab 6144. His research is focused on modelling, simulation and optimisation of energy systems, energy networks and multi-carrier energy systems.

ANTHONY MOURAUD has a PhD degree in Computer Science from the University Antilles/Guyane. He has 10 years of experience in the field of software development including skills such as artificial intelligence, prediction, neural networks, machine learning etc. Anthony holds digital and innovations at Lhyfe since 2021. Anthony was previously Deputy Director of CEA in Nantes, and head of the research team "Artificial Intelligence & Robotics" at CEA in Nantes leading a team of 12 research with mainly industry contracts in various areas: energy systems, robotics, agriculture, manufacturing, monitoring.