Hyperparameter Automation for Neural Networks in Solid Oxide Fuel Cell Systems

Internship report
Carl von Ossietzky Universität Oldenburg
Department of Computing Science

by REN Kevin

FISE 2024 Autonomous Robotics

ENSTA Bretagne

September 30, 2023

Supervised by

Prof. Dr.-Ing habil. Andreas Rauh

M.Sc. Marit Lahme

Acknowledgments. I am deeply grateful to Prof. Andreas Rauh and Marit Lahme for their guidance and support during the course of this project. Their expertise, and dedication have been invaluable in steering this research in the right direction. Their commitment to ensuring a thorough understanding of the subject matter, their patient guidance through complex challenges, and their willingness to provide insightful feedback have significantly enriched my learning experience.

Prof. Andreas Rauh's extensive knowledge in the field of Solid Oxide Fuel Cells (SOFCs) and his passion for research have been instrumental in shaping the depth and scope of this work. His guidance, mentorship, and willingness to engage in thought-provoking discussions have been a constant source of inspiration.

I would also like to express my gratitude to M.Sc. Marit Lahme for her continuous support, encouragement, and technical insights. Her expertise in neural networks and her patient guidance in navigating through intricate aspects of the research have been immensely beneficial.

Their encouragement to explore innovative ideas, their meticulous attention to detail, and their commitment to academic excellence have significantly contributed to the quality of this work. I am truly fortunate to have had the privilege of working with such accomplished individuals, and I extend my heartfelt appreciation for their impact on my academic journey.

Abstract

Traditional neural network models for complex dynamical systems lack explicit incorporation of structural engineering insights and the interconnectedness of various subprocesses that are related to the multiphysics nature of such systems. Consequently, these neural network models are often perceived as data-driven black boxes, in contrast to physically inspired equation-based representations, where appropriate parameters are identified in a more transparent manner. To bridge this gap, a new approach involving physics-inspired structuring of neural networks has been developed. This approach aims to enhance the modeling capabilities and understanding of complex systems by incorporating domain knowledge into the network's architecture without significant loss in terms of precision. This new approach has been applied to the thermal and electrochemical behaviour of high-temperature fuel cells. However, this initial model still requires human intervention to choose hyperparameters in order to obtain a fully meaningful network. The goal here is to make the system autonomous in determining appropriate hyperparameters.

Résumé

Les modèles traditionnels de réseaux de neurones pour les systèmes dynamiques complexes manquent d'intégration explicite des connaissances qu'il peut y avoir entre les différents sous-processus liés à la nature multiphysique de ces systèmes. Par conséquent, ces modèles de réseaux neuronaux sont souvent perçus comme des boîtes noires pilotées par les données, contrairement aux représentations basées sur des équations physiques où les paramètres appropriés sont identifiés de manière plus transparente. Pour combler cette différence, une nouvelle approche impliquant une structuration des réseaux de neurones inspirée de la physique a été développée. Cette approche vise à améliorer les capacités de modélisation et la compréhension des systèmes complexes en incorporant les connaissances du domaine dans l'architecture du réseau sans perte significative de précision. Cette nouvelle approche a été appliquée au comportement thermique et électrochimique des piles à combustible à haute température. Cependant, ce modèle initial nécessite encore l'intervention humaine pour choisir les hyperparamètres afin d'obtenir un réseau entièrement interprétable. L'objectif est de rendre le système autonome quant à la détermination des hyperparamètres.

Table of contents

1 Introduction

Optimizing the performance of solid oxide fuel cells (SOFC) represents an essential effort in the quest for increased energy efficiency and a reduction in battery production given the pollution caused by the latter. SOFCs are known for their notable efficiency metrics. They function in an ecologically responsible manner when the essential fuel, comprised of hydrogen and/or hydrocarbons, is sourced from renewable origins.

The integration of artificial neural networks has emerged as a promising avenue to model and control the intricate processes governing SOFC behavior. However, while it is possible to attain relatively swift outcomes with these neural networks, extracting meaningful physical information from them, especially in the case of multiphysical phenomena, proves to be exceedingly challenging. For this reason, researchers commonly regard neural networks as black-box models. Consequently, the motivation for physically inspired neural network structuring to model SOFC system behavior has grown [RKF+21] in order to increase the amount of information that can be extracted from the neural network.

Moreover the model obtained with a classic neural network may not be efficient enough to allow continuous monitoring of the system, which is necessary for control purposes. This can be accomplished through the application of nonlinear control methodologies, which are crafted using low-order yet adequately precise descriptions of dynamic systems [RSA15b, RSA15a, FRKA20]. Moreover given that these control approaches frequently require estimations of unmeasurable system states and perturbation variables, encompassing both transient heating phases and non-stationary high-temperature operations, researchers have highlighted the quasi-linearity present in dynamic models expressed by ordinary differential equations (ODEs), as well as the affine relationships linking control inputs and system states, as noteworthy advantages.

The opening segment Section 2 of this report provides a concise overview of solid oxide fuel cells and the intricacies associated with their modeling and control. We delve into the foundational theoretical principles that underlie our work.

In Section 3, is described the design of neural networks. We shed light on the conventional challenge posed by neural networks as "black-box" models, often difficult to interpret. In contrast, the approach presented takes a novel route by structuring the neural network to extract a tangible physical meaning.

In Section 4 is described how to take into account the different non-structural constraints, to adhere to physical considerations, ensuring a realistic depiction of system interactions.

Section 5 and Section 6 delve into the details of the development of algorithms to automate the search for optimal weights apparent in the cost function used during the training of the neural network, as well as number of hidden neurons in some specific layers. We introduce the application of simulated annealing for weights optimization, highlighting its efficacy in producing high-performance solutions. Additionally, we outline a clustering-based approach that aims to uncover sets of promising weight values, offering insights into potentially optimal neural network configurations.

Finally, conclusions and an outlook are presented in Section 8

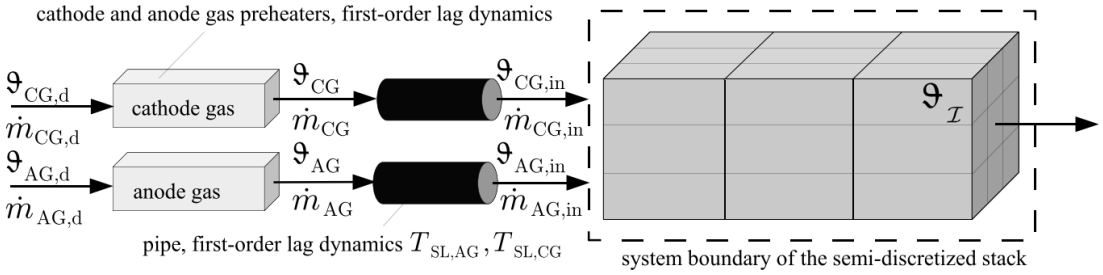
2 Equation-based modelling of the thermal behaviour of a SOFC stack

Generally, SOFCs are intricate systems composed of three primary components: the anode, the cathode, and the electrolyte. As the performance of SOFCs is enhanced by interconnecting more cells, the complexity of the modeling task escalates, owing to the growing number of distinct components, such as interconnections and seals [BA+17, Spi18].

SOFC systems and their constituent elements can be subject to modeling and analysis with diverse objectives and intentions, including the exploration of cell materials [TMK+16], examination of the impact of particle size in the anode microstructure [CX01], and the optimization of electrode microstructures [CYS+18]. In essence, the overarching goal of modeling a SOFC system is to enable its application in control and diagnostics.

2.1 Ordinary differential equation based model of the thermal behaviour of a SOFC stack

It has been shown that an equation-based model for the thermal behaviour of an SOFC system can be used to perform tasks including model-based parameter and state estimation, as well as control design for the heating and high-temperature reaction phases of the SOFC system [RSA15b, RSKA16]. The models for describing the thermal behaviour of a SOFC consists of two elements. The first part corresponds to stack modules and the second one corresponds to a preheater. The preheater is composed of an anode and a cathode as well as associated mass flow controllers (Figure 2.1). In this paper the focus will be limited to the stack.



$\vartheta_{AG,d}, \vartheta_{CG,d}$	- desired values for temperatures of anode and cathode gas
$\vartheta_{AG,in}, \vartheta_{CG,in}$	- temperature of anode and cathode gas in the inlet gas manifold
$\vartheta_{AG}, \vartheta_{CG}$	- temperatures of anode and cathode gas at the preheater outlet
$\dot{m}_{AG}, \dot{m}_{CG}$	- measured mass flows of anode and cathode gas at the preheater outlet
$\dot{m}_{AG,d}, \dot{m}_{CG,d}$	- desired mass flows of anode and cathode gas
T_{AG}, T_{CG}	- time constants of the first-order lag dynamics of the preheaters

Figure 2.1. Fuel stack module with gas preheaters for a parallel flow arrangement of cathode and anode gases [RKF+21].

In order to control the system in real-time, finite-dimensional sets of ODEs are chosen as a good compromise between modelling accuracy and computational complexity. Therefore, the fuel cell stack module is divided into $n_x = L\,M\,N$ elements of equal dimension. If

we denote L_L, L_M, L_N as the dimensions of the stack, each sub-cuboid has dimensions $l_L = L_L / L, l_M = L_M / M$ and $l_N = L_N / N$. To identify a sub-cuboid the indexing $\mathcal{I} \in [\![1, L]\!] \times [\![1, M]\!] \times [\![1, N]\!]$ is used (Figure 2.2.a). Hence, the temperature of the sub-cuboid indexed by I is given by

$$\dot{\vartheta}_{\mathcal{I}} = \frac{1}{c_{\mathcal{I}} m_{\mathcal{I}}} \left(\dot{Q}^{\mathcal{I}}_{\mathrm{HT}} + \sum_{G \in \{\mathrm{AG}, \mathrm{CG}\}} \dot{Q}^{\mathcal{I}}_{G, \mathcal{I}_j^-} + \dot{Q}^{\mathcal{I}}_{\mathrm{EL}} + \dot{Q}^{\mathcal{I}}_{R} \right) \tag{2.1}$$

where

- HT refers to heat tranfer
- G refers to enthalpy flow
- R refers to exothermic reaction enthalpy
- EL refers to omic losses
- see also Figure 2.2.b

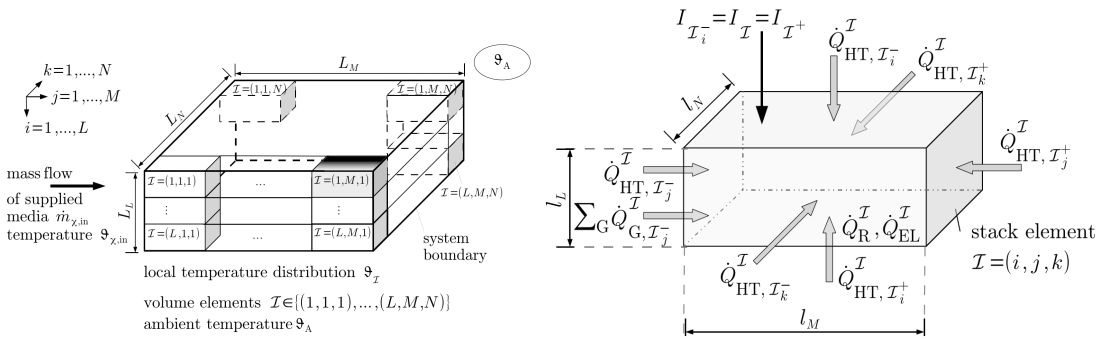


Figure 2.2. (a) Spatial semi-discretization of the fuel cell stack module (b) Local energy balance of the semi-discretized fuel cell stack module [RKF+21]

2.2 Quasi-linear structure

According to [IRK+19, RKA18] the previous non linear ODE model for the thermal system behaviour can be re-written into the quasi-linear, input-affine state equations. Let $\boldsymbol{x}$ be the state vector containing all finite volume element temperatures and $\boldsymbol{p}$ a parameter vector. The quasi-linear equation is then

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x}, \boldsymbol{p})\boldsymbol{x} + \boldsymbol{B}(\boldsymbol{x}, \boldsymbol{p})\boldsymbol{u} \tag{2.2}$$

This results from assuming the linearity in the stack temperatures for the heat transfer terms $\dot{Q}^{\mathcal{I}}_{\mathrm{HT}}$ and by considering the heat capacities of all gases and reaction enthalpies in $\dot{Q}^{\mathcal{I}}_{G,\mathcal{I}_j^-}$ and $\dot{Q}^{\mathcal{I}}_{R}$. Therefore, a multiplicative coupling between $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{p})$ and $\boldsymbol{x}$ is obtained and both $\boldsymbol{A}$ and the input matrix $\boldsymbol{B}$ depend on $\boldsymbol{x}$ and $\boldsymbol{p}$. $\boldsymbol{B}(\boldsymbol{x}, \boldsymbol{p})$ contains the ohmic losses $\dot{Q}^{\mathcal{I}}_{\mathrm{EL}}$.

Considering that $L = N = 1$ and $M = 3$ as done in [RKF+21], the matrix turns into

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & a_{23} \\ 0 & a_{32} & a_{33} \end{bmatrix} \quad \boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & 0 & 0 & b_{24} \\ b_{31} & 0 & 0 & b_{34} \end{bmatrix} \tag{2.3}$$

and

$$\boldsymbol{x} = \begin{bmatrix} \vartheta_{(1,1,1)} & \vartheta_{(1,2,1)} & \vartheta_{(1,3,1)} \end{bmatrix}^T$$

All inputs (control and disturbance variables, i.e., the ambient temperature, the anode and cathode gas inlet temperatures and the electric current) are summarized in the vector

$$\boldsymbol{u} = \begin{bmatrix} \vartheta_A & \vartheta_{\mathrm{AG,in}} & \vartheta_{\mathrm{CG,in}} & \frac{1}{3}I \end{bmatrix}^T$$

It is assumed that the current was homogeneous such as $I_{(1,1,1)} = I_{(1,2,1)} = I_{(1,3,1)} = \frac{1}{3}I$.

With in-depth structural analysis of this system model few properties are needed to reflect physically motivated stability properties [IRK+19, RKA18]

- all off-diagonal elements of the matrix $\boldsymbol{A}$ are positive (Metzler matrix)
- all diagonal elements of $\boldsymbol{A}$ are strictly negative
- all elements of $\boldsymbol{B}$ are positive
- from a thermal point of view $\vartheta_{\mathrm{CG,in}}$ is the only control input and other entries in $\boldsymbol{u}$ are disturbance inputs

3 Physically structured neural network modelling

It is desired to derive continuous-time network models. For this purpose, as mentionned in [RKF+21] static mapping between the current system states

$$\boldsymbol{x} = [x_1, ..., x_n]^T \tag{3.1}$$

as well as all measurable time-dependant control, input and disturbance variables

$$\boldsymbol{q} = [q_1, ..., q_m]^T \tag{3.2}$$

in the input layer of the neural network and the outputs of the network

$$\dot{\boldsymbol{x}} = [\dot{x}_1, ..., \dot{x}_n] \tag{3.3}$$

All of those vectors are low-pass filtered with the same time constant so undesirable phase shifts between the input-ouput relations are avoided. The system is then represented by

$$\dot{\boldsymbol{x}} = \boldsymbol{f}_{\mathbf{net}}(\boldsymbol{x}, \boldsymbol{q}) \tag{3.4}$$

instead of (2.1) and (2.2). However, this model does not distinguish between the different phenomena causing the temperature variations which is essential if we want to control the system. Such control procedures usually intend to manipulate the enthalpy flow of the cathode gas through adjustments in both the inlet temperature and the corresponding mass flow of the cathode gas. Both these parameters are components of the vector $\boldsymbol{q}$. By using such a control procedure, temperature fluctuations brought about by exothermic reaction enthalpies are mitigated.

3.1 Separation of heat transfer and exothermic reaction enthalpies

According to [RKF+21] the equation (3.4) can be divided into two parts, temperature variations due to exothermic reaction enthalpies and heat transfer phenomena

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_{\boldsymbol{R}} + \dot{\boldsymbol{x}}_{\mathbf{th}} = \boldsymbol{f}_{\mathbf{net},\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{q}) + \boldsymbol{f}_{\mathbf{net},\mathbf{th}}(\boldsymbol{x}, \boldsymbol{q})$$

Moreover it has been shown that $\boldsymbol{f}_{\mathbf{net},\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{q}) \equiv \mathbf{0}$ for vanishing currents $I \equiv 0$. Therefore , it can be inferred that during the heating phase, the values of $\boldsymbol{f}_{\mathbf{net},\boldsymbol{R}}$ represent directly generated disturbance heat flows and these can be compensated by a control process to keep the cell temperature within a certain range.

That leads to the following neural network structure

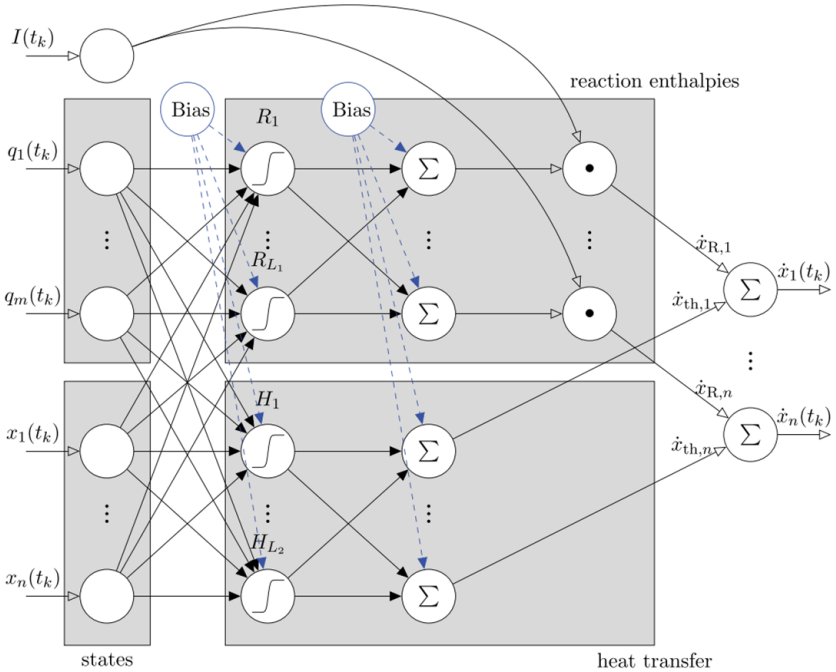


Figure 3.1. Network resulting from the separation of heat transfer and exothermic reaction enthalpies [RKF+21]

The connections indicated by the empty arrowheads represent connections for which the weights are determined a priori, as their configuration is known. They are therefore not modified during the training. This is the case, for example, for the summation between $\dot{\boldsymbol{x}}_{\boldsymbol{R}}$ and $\dot{\boldsymbol{x}}_{\mathbf{th}}$ is given by $a\,\dot{\boldsymbol{x}}_{\boldsymbol{R}} + b\,\dot{\boldsymbol{x}}_{\mathbf{th}}$ with $a = b = 1$. Connections where training are possible are indicated by filled arrowheads.

However, this structure has some issues, including overfitting that needs to be addressed. The solution proposed here involves using a quasi-linear modeling [RKF+21]

$$\dot{\boldsymbol{x}}_{\mathbf{th}} = \boldsymbol{A}(\boldsymbol{x}, \boldsymbol{q})\boldsymbol{x} + \boldsymbol{b}(\boldsymbol{x}, \boldsymbol{q})\vartheta_{\mathrm{CG,in}} + \boldsymbol{d}(\boldsymbol{x}, \boldsymbol{q}) \tag{3.5}$$

Just like in the equation-based approach (2.2), a state-dependent system matrix $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{q})$ and a state-dependent input vector $\boldsymbol{b}(\boldsymbol{x}, \boldsymbol{q})$ are introduced. The latter is linearly coupled with the cathode gas inlet temperature $\vartheta_{\mathrm{CG,in}}$ as a control variable. All terms that do not fit into this structure, such as dependencies on the ambient temperature ϑ_A and the anode gas inlet temperature $\vartheta_{\mathrm{AG,in}}$, are captured by the disturbance term $\boldsymbol{d}(\boldsymbol{x}, \boldsymbol{q})$. The $\dot{\boldsymbol{x}}_{\boldsymbol{R}}$ bloc remains the same [RKF+21].

To represents the matrix product $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{q})\boldsymbol{x}$, the linear subnetwork output in Figure 3.2 encapsulates all matrix elements using a column-wise notation. The weights associated with the subsequent layers of multiplication and addition are then set to either 1 or 0 in order to achieve the desired product. The block performing the product $\boldsymbol{b}(\boldsymbol{x}, \boldsymbol{q})\vartheta_{\mathrm{CG,in}}$ is parameterized in a similar manner. All these weights are fixed during the training stage [RKF+21].

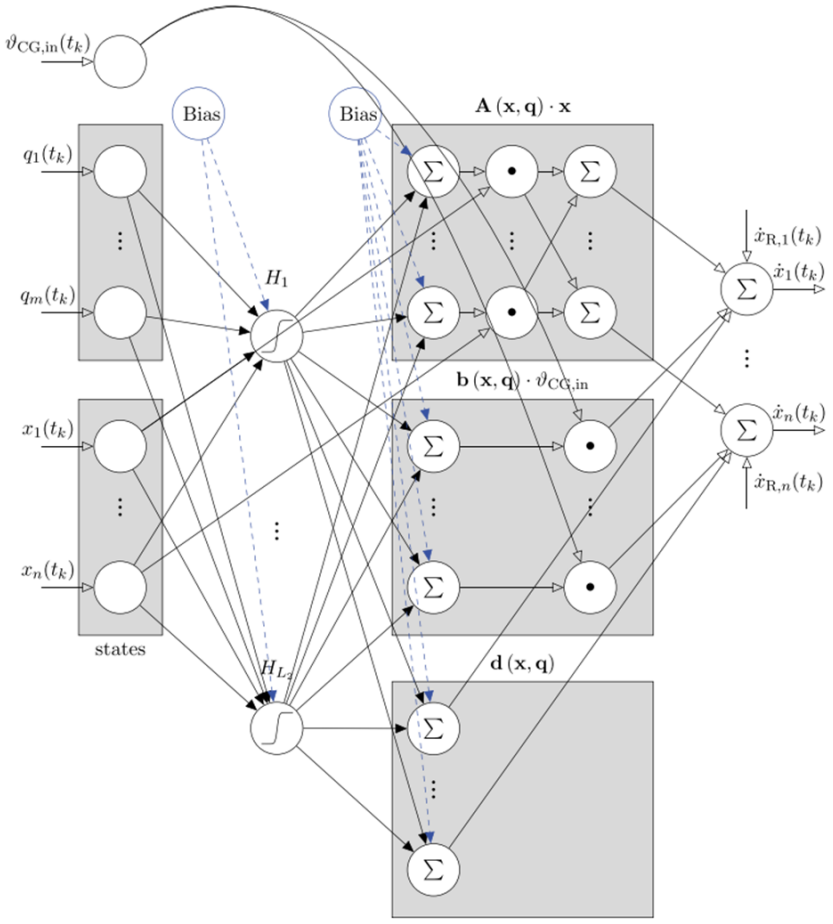


Figure 3.2. Quasi-linear representation of the heat transfer subnetwork

In contrast to the analytical model of the system (2.3), no constraints are enforced on $\boldsymbol{A}(\boldsymbol{x}, \boldsymbol{q})$ and $\boldsymbol{b}(\boldsymbol{x}, \boldsymbol{q})$ during the neural network training. From a modeling perspective, this signifies a level of freedom that the training algorithm can exploit, allowing it to consider

interactions not only among directly adjacent elements but also those originating from more distant elements. This is similar to higher-order spatial discretization techniques, commonly used for numerically solving partial differential equations [Gus07].

The Bayesian regularization backpropagation algorithm and mean squared error (MSE) regarding the derivatives of the state variables $\mathrm{MSE}_{\mathrm{IN}}$ will be employed for training the neural network.

$$\mathrm{MSE}_{\mathrm{IN}} = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x_M} - \boldsymbol{x})_i^2 = \frac{1}{n}\sum_{i=1}^{n} (\dot{\vartheta}_{M,i} - \dot{\vartheta}_i)^2 \tag{3.6}$$

where the subscript M indicates that it is a measured value the absence of this subscript indicates a predicted value.

Remark 3.1. The hidden layer H introduces nonlinearity into the calculation of the coefficients of the matrix $\boldsymbol{A(x, q)}$ and the vectors $\boldsymbol{b(x, q)}$ and $\boldsymbol{d(x, q)}$, aiming to better capture the physical phenomena.

4 Incorporating Physical Constraints into the Neural Network

Throughout the rest of the paper, we will use either T_n or $T^{(n)}$ interchangeably to denote the value of a variable T at the n-th iteration.

As mentionned in [Eil23] to truly have a physical meaning, several constraints must be applied. The matrix $\boldsymbol{A(x, q)}$ must be Metzler, meaning that all diagonal coefficients must be strictly negative and off-diagonal coefficients must be positive. The coefficients of the vector $\boldsymbol{b(x, q)}$ must must be positive. Moreover high temperature variations within the battery are not feasible. Thus, $\dot{\boldsymbol{x}}$ must be bounded. The final condition concerns the system stability, the system is asymptotically stable if all real parts of the eigenvalues of $\boldsymbol{A(x, q)}$ are strictly negatives.

4.1 Constraints on signs

To implement the constraints on matrix $\boldsymbol{A(x, q)}$ within the neural network, new outputs have been added [Eil23].

Let's revisit the neural network depicted in Figure 3.2 and focus on the subnetwork corresponding to the product $\boldsymbol{A(x, q)x}$. The first column consists of n^2 summations that calculate the n^2 coefficients of matrix $\boldsymbol{A(x, q)}$ (these coefficients are circled in green in Figure 4.1). In order to enforce the signs of these coefficients, they will be associated with outputs $\left(x_A^{(i)}\right)_{1 \leqslant i \leqslant n^2}$ (see Remark 4.1 for more details), as illustrated in Figure 4.1. The weights of the new connections will be set to 1, and the activation function used for these outputs will be the hyperbolic tangent

$$\begin{array}{rccc} \tanh\colon & \mathbb{R} & \to &]-1,1[\\ & x & \mapsto & \dfrac{1-e^{-2x}}{1+e^{-2x}} \end{array}$$

This activation function will serve to determine the signs of the coefficients while avoiding coefficients that are indeed with the correct sign but too close to 0. This could increase the chances of these coefficients changing sign and taking on an unintended sign. In our case $\boldsymbol{A(x,q)}$ must be Metzler. So, a matrix $\boldsymbol{A_d}$ such as

$$\forall (i,j) \in [\![1,n]\!]^2 \quad \begin{cases} i=j \Rightarrow a_d^{(ij)} < 0 \\ i \neq j \Rightarrow a_d^{(ij)} \geqslant 0 \end{cases}$$

is created and will be used as target values during the training.

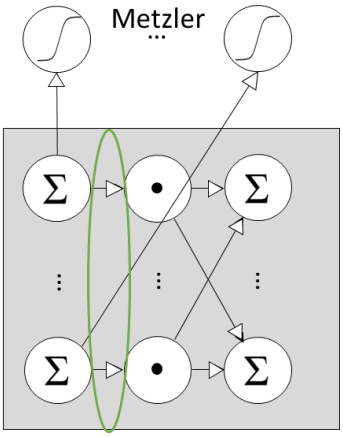


Figure 4.1. Extension of the $\boldsymbol{A(x,q)x}$ subnetwork for Metzler character in matrix $\boldsymbol{A}$

The error related to this new output block is defined by

$$\text{MSE}_A = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i=j}^{n} \left(\tanh(a^{(ij)}) - a_d^{(ij)}\right)^2 \tag{4.1}$$

***Remark* 4.1.** *Let f_c be the function that associates, for any matrix $\boldsymbol{M}$ of arbitrary size $(n,m) \in (\mathbb{N}^*)^2$, the vector $\boldsymbol{v_M}$ such as*

$$\forall (i,j) \in [\![1,n]\!] \times [\![1,m]\!] \quad m^{(ij)} = v_M^{(m(i-1)+j)}$$

If $m=1$, $\boldsymbol{M}$ is a vector and $f_c = \text{Id}$. In the neural network, the matrix $\boldsymbol{A}$ is represented by the vecteur $\boldsymbol{v_A} = \boldsymbol{f_c(A)}$. The outputs $\left(x_A^{(i)}\right)_{1 \leqslant i \leqslant n^2}$ corresponds to $\left(\tanh\left(v_A^{(i)}\right)\right)_{1 \leqslant i \leqslant n^2}$ and the training data $\left(y_A^{(i)}\right)_{1 \leqslant i \leqslant n^2}$ is equal to $(f_c(A_d)_i)_{1 \leqslant i \leqslant n^2}$ and MSE_A is calculated as

$$\text{MSE}_A = \frac{1}{n^2} \sum_{i=1}^{n^2} \left(x_A^{(i)} - y_A^{(i)}\right)^2.$$

Similary vector $\boldsymbol{b(x,q)}$ must be positive. Therefore, we add a third output block related to the first sum column (see Figure 4.2). These outputs again use the hyperbolic tangent activation function, and the target vector is $\boldsymbol{b_d} = (1)_{1 \leqslant i \leqslant n}$. The error related to this output block is

$$\text{MSE}_B = \frac{1}{n} \sum_{i=1}^{n} \left(\tanh(b^{(i)}) - b_d^{(i)}\right)^2 \tag{4.2}$$

Using the notation introduced in Remark 4.1 the relation become

$$\mathrm{MSE}_B = \frac{1}{n}\sum_{i=1}^{n} \left(x_b^{(i)} - y_b^{(i)}\right)^2 \tag{4.3}$$

Just like with the case above, the weights of the new connections are set to 1.

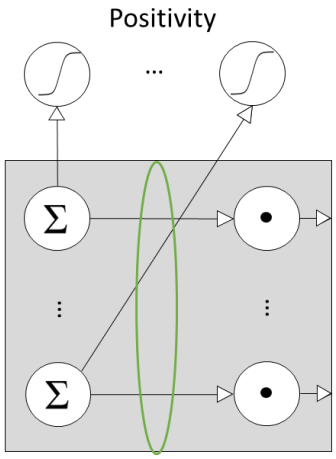


Figure 4.2. Extension of the $\boldsymbol{b}(\boldsymbol{x}, \boldsymbol{q})\vartheta_{\mathrm{CG,in}}$ subnetwork for the vector $\boldsymbol{b}$ positivity

4.2 Temporal variation rates of state variables constraint

In addition to the previous extensions, this section presents the extension that constrains the variation of temperature over time is reasonable to prevent thermal stress and material degradation [RKF+21].

Consider two variables $(\boldsymbol{g}_{\mathbf{Temp1}}, \boldsymbol{g}_{\mathbf{Temp2}}) \in (\mathbb{R}_+^n)^2$ such as

$$\forall i \in [\![1, n]\!] \quad -g_{\mathrm{Temp1}}^{(i)} < \dot{x}_i < g_{\mathrm{Temp2}}^{(i)} \tag{4.4}$$

To be considered, these variables will be linked to two new inputs in the neural network.

$$\forall i \in [\![1, n]\!] \quad \begin{cases} \dot{x}_i - g_{\mathrm{Temp2}}^{(i)} < 0 \\ -\dot{x}_i - g_{\mathrm{Temp1}}^{(i)} < 0 \end{cases} \tag{4.5}$$

The newly added ouputs $\left(x_{\mathrm{Temp1}}^{(i)}\right)_{1 \leqslant i \leqslant n}$ and $\left(x_{\mathrm{Temp2}}^{(i)}\right)_{1 \leqslant i \leqslant n}$ here correspond to

$$\forall i \in [\![1, n]\!] \quad \begin{cases} x_{\mathrm{Temp1}}^{(i)} = \tanh\left(-\dot{x}_i - g_{\mathrm{Temp1}}^{(i)}\right) \\ x_{\mathrm{Temp2}}^{(i)} = \tanh\left(\dot{x}_i - g_{\mathrm{Temp2}}^{(i)}\right) \end{cases} \tag{4.6}$$

The target vectors used for training is the same for all these outputs and is $\boldsymbol{y}_{\mathbf{Temp}} = (-1)_{1 \leqslant i \leqslant n}$.

Hence, the errors related to this extension are given by

$$\begin{aligned} \mathrm{MSE}_{\mathrm{Temp1}} &= \frac{1}{n}\sum_{i=1}^{n} \left(x_{\mathrm{Temp1}}^{(i)} - y_{\mathrm{Temp}}^{(i)}\right)^2 \\ \mathrm{MSE}_{\mathrm{Temp2}} &= \frac{1}{n}\sum_{i=1}^{n} \left(x_{\mathrm{Temp2}}^{(i)} - y_{\mathrm{Temp}}^{(i)}\right)^2 \end{aligned} \tag{4.7}$$

The weights of the connections introduced here are correctly set to 1 or -1.

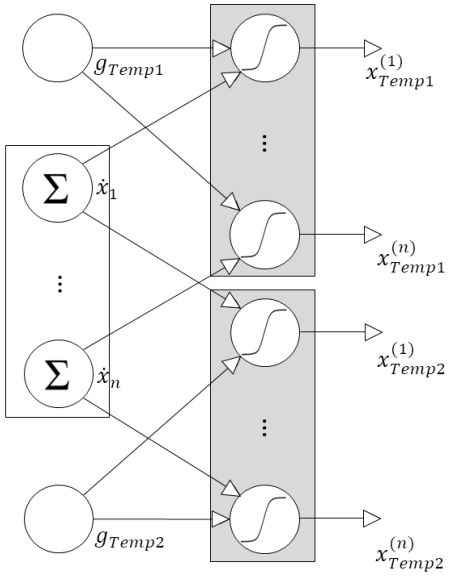


Figure 4.3. Extension that bounds $\dot{\boldsymbol{x}}$

4.3 Stability constraint

A final extension [Eil23] is added which achieves continuous asymptotic stability of the system i.e. all eigenvalues of the matrix $\boldsymbol{A(x,q)}$ must have negative real parts. To impose this, the Gershgorin theorem is used.

For $i \in [\![1,n]\!]$ let R_i be

$$R_i = \sum_{\substack{1 \leqslant j \leqslant n \\ j \neq i}} |a_{ij}|$$

Let $D(a_{ii}, R_i)$ be the disk defined by

$$D(a_{ii}, R_i) = \{z \in \mathbb{C}, |a_{ii} - z| \leqslant R_i\}$$

Theorem 4.1. *(The Gershgorin circle theorem) Every eigenvalue of* $\boldsymbol{A}$ *belongs to at least one of the Gershgorin disks.*

Hence, if for all i belonging to $[\![1,n]\!]$ the inequality $\mathrm{Re}(a_{ii}) + R_i < 0$ is respected then we can ensure that all the eigenvalues of $\boldsymbol{A(x,q)}$ have negative real parts.

Let consider that the matrix $\boldsymbol{A(x,q)}$ is Metzler[4.1]. Therefore

$$R_i = \sum_{j \neq i} |a_{ij}| = \sum_{j \neq i} a_{ij}$$

and $a_i < 0$. Hence

$$\mathrm{Re}(a_{ii}) + R_i < 0 \Leftrightarrow \sum_{1 \leqslant j \leqslant n} a_{ij} < 0$$

To implement this in the neural network, a new hidden layer is added to the $\boldsymbol{Ax}$ subnetwork. it will calculate the sum $\sum_{1 \leqslant j \leqslant n} a_{ij}$ for each i belonging to $[\![1,n]\!]$. The outputs associated with these sums are

$$x^{(i)}_{\mathrm{St}} = \tanh\left(\sum_{1 \leqslant j \leqslant n} a_{ij} \right) \tag{4.8}$$

and the target vector used for the training is $\boldsymbol{y_{St}} = (-1)_{1 \leqslant i \leqslant n}$.

4.1. This condition is necessary in all cases so assuming this is not absurd

4.4 Error correction of neural network

Let $\mathcal{J} = \{A, B, \text{Temp1}, \text{Temp2}, \text{St}\}$. As there are several output blocks, each block can be assigned a weight so that the cost function used to train the neural network is given by

$$\text{MSE}_T = w_{\text{IN}} \text{MSE}_{\text{IN}} + \sum_{j \in \mathcal{J}} w_j \text{MSE}_j \tag{4.9}$$

Due to the fact that the arg min of the MSE function is what matters, the weight w_{IN} can be fixed to 1. Indeed

$$\forall \lambda > 0 \quad \arg\min\left(\lambda \text{MSE}_T\right) = \arg\min\left(\text{MSE}_T\right) = \arg\min\left(\frac{1}{w_{\text{IN}}} \text{MSE}_T\right)$$

5 Error weights

The purpose of this part is to determine the error weights (4.9) that will be used during the training in order to robustly fulfill the conditions

- $\boldsymbol{A(x, q)}$ have to meet Metzler properties
- $\boldsymbol{b(x, q)}$ must be non-negative
- temperature change rate must be bounded (e.g. between $-5K/s$ and $+5K/s$)
- All the eigenvalues of $\boldsymbol{A(x, q)}$ have negative real parts

5.1 General process

The method employed for the error weigths automation is simulated annealing. It is a statistical method inspired by metallurgy that aims to find the global minimum of a function. By analogy with the physical process, the function to minimize is called the system's energy E. We also introduce a fictitious parameter, the system's temperature T. The purpose of this variable will be explained later.

Starting from a given state of the system, by modifying it, another state is obtained. Either this new state improves the criterion to optimize - that means the system's energy has decreased - or it worsens the criterion. Accepting a state that improves the criterion tends to search for the optimum in the vicinity of the initial state. Accepting a "bad" state allows us to explore a larger part of the state space and tends to prevent from getting trapped too quickly in the search for a local optimum.

This method has the advantage of not requiring various properties on the function to be minimized (such as continuity, differentiability, etc.) and does not settle for the first local minimum found.

Theoretical studies have shown that under certain conditions, simulated annealing can converge to the global minimum; however, this will not be our case. Nevertheless, it will manage to find an acceptable solution.

5.2 Algorithm

The initial state can be randomly selected from the set of all possible states. This state is associated with an energy $E = E_0$. Additionally, an initially high temperature $T = T_0$ is selected in a completely arbitrary manner.

At each iteration of the algorithm, the state is modified. This modification results in a variation ΔE of the system's energy. If this variation is negative (i.e., it reduces the system's energy), it is applied to the current state. Otherwise, it is accepted with certain probability. For example, the probability can be $e^{\left(-\frac{\Delta E}{T}\right)}$. The choice of using the exponential function for the probability is known as the Metropolis rule.

There are several ways to reduce the temperature. Here, we have chosen to decrease the temperature at each iteration following the formula : $T_{n+1} = \lambda T_n$ where $0 \ll \lambda < 1$.

5.2.1 Pseudo-code

Let e be the energy associated with state $\boldsymbol{x}$, and m be the minimum energy found, which is associated with state $\boldsymbol{g}$. Let E be the function such as $e = E(\boldsymbol{x})$. Here is the pseudo-code

Algorithm 5.1

$x := x0$
$g := x0$
$e := E(x)$
$m := E(g)$
$k := 0$
while $k < k\max$ **and** $e > e\max$
    $xn := \text{neighbor}(x)$
    $en := E(xn)$
    $dE := en - e$
    if $en < e$ **or** $\text{rand}() < \mathbb{P}(dE, T)$
        $s := sn$
        $e := en$
    end if
    if $e < m$
        $g := s$
        $m := e$
    end if
    $k = k + 1$
    $T = \lambda T$
end while

5.2.2 Modifications

In our case, the state is represented by the error weights $\boldsymbol{e_w} = (1, w_a, w_b, w_{\text{Temp1}}, w_{\text{Temp2}}, w_{\text{St}})$ and the error is given by

$$E(\boldsymbol{e_w}) = \sqrt{\frac{1}{N_s}(\text{MSE}_A + \text{MSE}_B + \text{MSE}_{\text{Temp1}} + \text{MSE}_{\text{Temp2}} + \text{MSE}_{\text{St}})} \tag{5.1}$$

where n is the dimension of $\boldsymbol{x}$ (3.5) and N_s the number of sampling.

The first element of $\boldsymbol{e_w}$ is kept constant equal to 1 then the possible states belong to $\{1\} \times (\mathbb{R}_+^*)^5$ but in order to increase the speed of the program the possible states are restricted to the set $E_w = \{1\} \times [0.1, 10]^5$.

In the function E, the network is trained in the same way as in Algorithm 5.3. The obtained outputs are compared with the desired outputs to calculate the errors used in Equation (5.1).

The neighbor function depends on the iteration number k and returns a random vector y such as :

$$\begin{cases} y_1 = 1 \\ \forall i \in [\![2, 6]\!] \quad y_i \in [x_i - v\alpha(k), x_i + \alpha(k)\,] \cap E_w \end{cases} \tag{5.2}$$

more precisely $\alpha(k) = A\left(\frac{k_{\max} - k}{k_{\max}} + b\right)$ where $A \in \mathbb{R}_+$, $b \in \mathbb{R}_+$ and $1 < v \ll 2$. As result, the function neighbor slightly tends to descrease x rather than increase it. This is explained by the fact that we want the temperature inside the stack to be predicted accurately.

Therefore, the weight of the error associated with temperature variation should not be too low compared to the other error weights that provide physical meaning to the neural network.

Other modifications were made so that the algorithm could adapt to the code. Let a fixed neural network be given. The termination condition becomes $k > k\max = 50$ **or** $E(\boldsymbol{e_w}) < e_{\max}$ **and** the first two conditions have been met at least n' times for n' different trainings starting from the exactly the same neural network[5.1]. This is done with the aim of making the error weight given by the simulated annealing function more robust. For this purpose, the function $E(xn)$ will actually return a boolean "success" in addition to the error en. This boolean indicates whether or not the conditions have been met throughout the operating period. Then the probability $\mathbb{P}$ mentionned in Section 5.2 is defined by $e^{-\frac{\text{en}-e}{T}}$. The function returns the state $\boldsymbol{e_w}$, the temperature T but also the trained neural network.

Finally, here is the pseudo-code of the simulated annealing used.

Algorithm 5.2

Entries : $(x, \text{neighbor}, net0, k\max, T_0, X, Y)$
$(g, T) = (x, T_0)$
$e, success = E(x, net0, X, Y)$
$m = e$
$k = 0$
count $= 0$
while $k < k\max$ **and** $(e > e\max$ **or** $count < n')$
  if $success =$ false
    $xn = \text{neighbor}(x)$
  end if
  $en, success = E(xn, \text{net}, X, Y)$
  if $success$
    $count = count + 1$
  else
    $count = 0$
  end if
  if $en < e$ **or** rand() $< \mathbb{P}(en - e, T)$
    $s = sn$
    $e = en$
  end if
  if $e < m$ **or** count $\geqslant n'$
    $g = s$
    $m = e$
  end if
  $k = k + 1$
  $T = \lambda T$
end while

5.3 Implementation

It is recalled that the neural network trains on data retrieved from the stack through 8 hours of operation. In the initial code, N_{tr} trainings were performed to obtain the best neural network among the N_{tr} networks. When the error given by the simulation is low

5.1. Due to the stochastic nature of the trainings, the results obtained after the different trainings will be different, even when starting from the same initial network.

enough, the program can be stopped before the N_{tr} simulations. Now, we want this code to fulfill the first two mentioned criteria. For this purpose, whenever these conditions are not met simulated annealing function will be called, which will return a trained network that should satisfy the required conditions.

Thereby a certain fraction of the N_{tr} simulations meet the required conditions, and then we can choose the best among this subset.

5.3.1 Main loop

Note that in all the subsequent pseudo-codes, along with their accompanying explanations, may not precisely match the actual code. Some lines have been omitted, and others have been simplified. However, the code's principle remains the same. This has been done to make the explanation as clear as possible.

Firstly, the $\boldsymbol{X_{\text{sim}}}$ vector used here is not the state vector as described in the previous sections but the input vector of the neural network

$$\boldsymbol{X_{\text{sim}}} = \left(x_{\text{sim}}^{(i)}\right)_{1\leqslant i\leqslant n+m} = [x_1, ..., x_n, q_1, ..., q_m]^T \tag{5.3}$$

Using this vector enables us to analyze the neural network's performance more easily.

At the beginning of each iteration, a simulation[5.2] is conducted using a random trained network (Algorithm 5.3). The simulation function has been modified to be able to return two errors

- errorRate corresponds to the period during which one of the two conditions in A and B has not been met divided by the total duration of the simulation
- perfErrorRate corresponds to the period during which any of the criteria has not been met divided by the total duration of the simulation

If the errorRate given by the initial training is strictly positive, simulated annealing function will be used. Then, the accuracy of the simulation is calculated by determining the Root Mean Square Error (RMSE) between the predicted state vector $\boldsymbol{x}$, which corresponds to $\left(x_{\text{sim}}^{(i)}\right)_{1\leqslant i\leqslant n}$ and actual state vector.

At the end of each iteration, this final neural network is saved, along with the various errors and the simulation.

T_0 corresponds to the initial temperature given to the simulated annealing function and T_{sim} the current temperature. To take into account the previous calls to the simulated annealing function, tis function returns $\boldsymbol{e_w}$ and the temperature T_{sim}, which will be passed as parameters in the next call to it. But before that, in the next iteration, the temperature T_{sim} will be modified as follows

$$T_{\text{sim}}^{(n+1)} = \min\left(T_{\text{sim}}^{(n)}(0.9 + \text{errorRate}), T_0\right) \tag{5.4}$$

This way, the more consecutive good results we achieve with the same $\boldsymbol{e_w}$ vector (errorRate $<$ 0.1), the more reliable it becomes. Consequently, there is a higher chance of converging towards a minimum around $\boldsymbol{e_w}$ as T_{sim} decreases. The minimal error found by the function is not returned, as it varies significantly with the provided neural network. With each new iteration, this error is highly modified, making the previous error no longer representative.

5.2. See [RKF+21] for more details.

Algorithm 5.3

init $ew, ew_{\max} T_0$
$T_{\text{sim}} = T_0$
while *true*
    $runs = runs + 1$
    $net = \text{init}(net)$
    $net_0 = net$
    $net, tr = \text{train}(net, X, Y, ew)$
    $X_{\text{sim}}, errorRate, perfErrorRate = \text{simulation}()$
    $T_{\text{sim}} = T_{\text{sim}}(0.9 + errorRate)$
    $\text{T}_{\text{sim}} = \min(T_{\text{sim}}, T_0)$

    if $errorRate > 0$
        $e_w, T_{\text{sim}}, net, tr = \text{simannealing}(ew, \text{neighbor}, net_0, N_{\max}, T_{\text{sim}}, X, Y)$
        $X_{\text{sim}}, errorRate, perfErrorRate = \text{simulation}()$
    end if
    $errorRate_{\text{all}} = \text{concatenate}(errorRate_{\text{all}}, errorRate)$
    $perfErrorRate_{\text{all}} = \text{concatenate}(perfErrorRate_{\text{all}}, perfErrorRate)$
    $net_{\text{all}}[runs] = \text{concatenate}(net_{\text{all}}, net)$
    $sim_{\text{all}}[runs] = X_{\text{sim}}$
    $guete_{\text{sim}}[\text{runs}] = \text{RMSE}(X_{\text{sim}}(1{:}\,n))$

    if $\min(guete_{\text{sim}}) < 3$ or $runs > N_{\text{tr}}$
        break
    end if
end while

At the very end of the code, we select the neural network that achieved the best result, meaning the one with the least error in predicting the temperatures within the stack, and for which the conditions on $\boldsymbol{A}$ and $\boldsymbol{b}$ were met throughout the simulation, and the other conditions were met 99.9% of the time.

5.4 Silhouette coefficient

Given the results obtained in Section 7 it is likely that multiple satisfactory $\boldsymbol{e_w}$ vectors can be determined. Therefore, clustering is being considered to identify these vectors. Clustering is a data analysis technique that involves grouping similar data points together into clusters or subsets while keeping dissimilar points in separate clusters. The goal is to identify patterns, relationships, or structures within the data without any prior knowledge of specific groupings. It is achieved by measuring the similarity or dissimilarity between data points, often using distance metrics, and then assigning data points to clusters based on these measurements. The clustering algorithm that is used is the k-means algorithm. In order to evaluate the clustering, the silhouette coefficient is introduced.

Consider that the k-means algorithm has been applied to a set of points in E. We denote C_k as the k-th cluster, n_k as the number of points in C_k, and d as the function that calculates the Euclidean distance between two points. Let k be an interger and $\boldsymbol{x}$ (which corresponds here to $\boldsymbol{e_w}$) be a point in C_k. We define the functions

– $a(\boldsymbol{x})$ corresponds to the average distance between a data point and all other data points within the same cluster. Essentially, it quantifies how similar a data point

is to its fellow members within its assigned cluster.

$$a(\boldsymbol{x}) = \frac{1}{n_k - 1} \sum_{\boldsymbol{u} \in C_k, \boldsymbol{u} \neq \boldsymbol{x}} d(\boldsymbol{x}, \boldsymbol{u})$$

– $b(\boldsymbol{x})$ corresponds to the smallest average distance between the data point and all data points belonging to different clusters. It assesses how dissimilar a data point is to data points within neighboring clusters.

$$b(\boldsymbol{x}) = \min_{l \neq k} \frac{1}{n_l - 1} \sum_{\boldsymbol{u} \in C_l} d(\boldsymbol{x}, \boldsymbol{u})$$

The silhouette score of $\boldsymbol{x}$ is calculated as

$$s(\boldsymbol{x}) = \frac{b(\boldsymbol{x}) - a(\boldsymbol{x})}{\max\left(a(\boldsymbol{x}), b(\boldsymbol{x})\right)}$$

The silhouette score ranges from -1 to 1, where

– A high value indicates that the data point is well matched to its own cluster and poorly matched to neighboring clusters.

– A value near 0 indicates that the data point is on or very close to the decision boundary between two neighboring clusters.

– A low value indicates that the data point may have been assigned to the wrong cluster.

The silhouette plot (Figure 5.1) displays these silhouette scores for each data point in a cluster. Each bar corresponds to a point. The y-axis indicates which cluster the bar belongs to. The length of each bar represents the silhouette score of the corresponding data point.

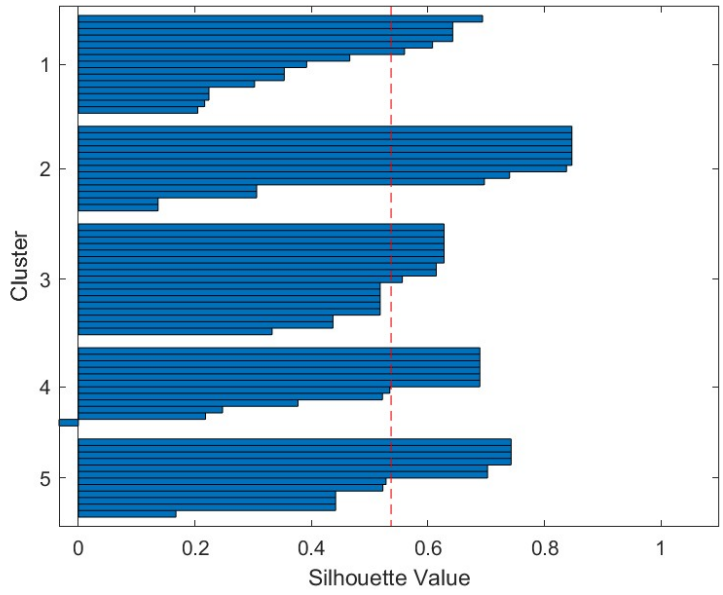


Figure 5.1. Silhouette plot

Depending on the total number of clusters, the outcomes can vary significantly between different runs of the k-means algorithm. Highly unstable results for the same total number of clusters indicate that this number is not suitable for our data. For example, when considering 2 clusters for the set of points shown in Figure 5.2, we can obtain the two displayed configurations. Because there are two, or even three, possible clustering scenarios, the result is unstable. This means that different runs of the k-means algorithm can yield different cluster assignments.

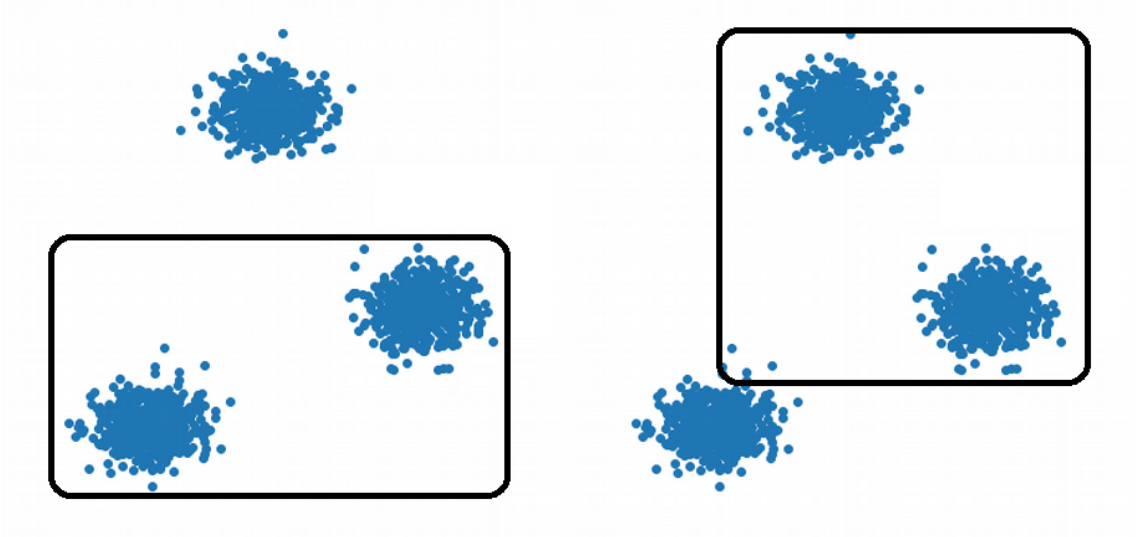


Figure 5.2. Instability of k-means with 2 clusters

6 Hidden layers neurons

In this section, we attempt to find an automatic way of determining the numbers of neurons L_1 and L_2 in the hidden layers R and H. To achieve this, the method used will be very similar to simulated annealing. These hidden layers are the only ones that can be modified since the other layers serve specific calculations. The goal is to find the optimal number of hidden neurons within these layers. Therefore, to avoid overfitting and reduce computation time when using the trained network, the number of hidden neurons should be kept low but still high enough not to compromise prediction quality.

6.1 General process

The state vector in this case is no longer $\boldsymbol{e_w}$ but rather $\boldsymbol{nv} = (L_1, L_2)$. The objective function for this second simulated annealing function is the median prediction error given over the last n_{count} simulations with the same $\boldsymbol{nv}$. However, if for the same pair (L_1, L_2) the failure rate exceeds a certain percentage, i.e., the conditions on $\boldsymbol{A}$ and $\boldsymbol{b}$ are not met for a certain percentage of simulations, then this pair is rejected. Since the conditions on $\boldsymbol{A}$ and $\boldsymbol{b}$ depend only on L_2, if the failure rate is too high, it implies that the number of neurons in the hidden layer H is too low. Therefore, we can set a minimum value for L_2, which will be the last value of L_2 for which a failure occurred.

6.2 Algorithm

First, let's focus on Algorithm 6.1. This algorithm is placed at the beginning of the while loop of Algorithm 5.3. The first if statement is used to check the failure rate mentioned earlier. If it exceeds 20%, we redefine the new minimum value for L_2. Initially, the minimums

for L_1 and L_2 were set to 0. If E_{nv} is the set of all possible states for $\boldsymbol{nv}$, the neigbhor function returns a vector $\boldsymbol{y}$ such as

$$\forall i \in [\![1,2]\!] \quad y_i \in [x_i - \upsilon_{nv}\alpha_{nv}(k), x_i + \alpha_{nv}(k)\,] \cap E_{nv}$$

After modifying the number of hidden neurons T_{sim} is reset to T_0 and the counters are reset to 0. The variables *count* and *failcount* are used to calculate the failure rate, such that

$$failrate = failcount\,/\,count$$

The second if statement is used when the failure rate is below 20%. This indicates a successful attempt. The median of the last n_{count} errors is calculated and stored in an array that records all the median errors. The subsequent "**if** $en < e$ **or** rand() $< \mathbb{P}(en - e, T)$" statement is explained in Section 5.1. Following this, similar to the first if statement, a new state for L_1 and L_2 are modified, T_{sim} and the counters are reset to their initial values.

Algorithm 6.1

if count $< n_{\text{count}}$ **and** failcount $\geqslant 0.2 n_{\text{count}}$
  $nv_{\min}[2] = nvn[2]$
  $nvn = \text{neigbhor}(nv)$
  $nvn = \max(nvn, nv_{\min})$
  $(L_1, L_2) = nvn$
  $T_{\text{sim}} = T_0$
  $count = 0$
  $failcount = 0$
end if
if $count \geqslant n_{\text{count}}$
  $n = \text{size}(guete_{\text{sim}})$
  $M = guete_{\text{sim}}[n - n_{\text{count}} + 1, n]$
  $en = \text{median}(M)$
  $median_{\text{all}} = \text{concatenate}(median_{\text{all}}, en)$
  if $en < e$ **or** rand() $< \mathbb{P}(en - e, T_{\text{nv}})$
    $e = en$
    $nv = nvn$
  end if
  $nvn = \text{neigbhor}(nv)$
  $nvn = \max(nvn, nv_{\min})$
  $(L_1, L_2) = nvn$
  $T_{\text{sim}} = T_0$
  $count = 0$
  $failcount = 0$
end if

The second algorithm of this section is placed juste after the first if statement in Algorithm 5.3. This algorithm is responsible for incrementing the counters.

Algorithm 6.2

if $errorRate > 0$
  $failcount = failcount + 1$
end if
$count = count + 1$
$T_{\text{nv}} = 0.95\, T_{\text{nv}}$

7 Training and results

7.1 Training

The training and subsequent numerical assessment of all networks were grounded in the collection of actual data obtained from a SOFC testing facility at the University of Rostock. This data collection spanned slightly over eight hours, encompassing the phase of high-temperature reaction during the final 1.5 hours. The utilization of these training datasets, which include a considerable number of data points even during electrochemically inactive phases, offers the distinct advantage of accurately characterizing the impact of cathode gas enthalpy fluxes. This detailed understanding paves the way for the development of precise temperature control strategies, aligning with the objectives outlined in introduction of Section 3. The experimental data employed in this section, encompassing details about all segment temperatures, gas mass flow rates, stack inlet temperatures, as well as current and voltage at the terminals, were sampled at a temporal resolution of 100 ms, resulting in approximatively 300,000 data points. Subsequently, the data is filtered using a first order low-pass filter with a edge frequency of 1Hz. This frequency is also employed for a low-pass filtered derivative estimation after its automatic numerical discretization using the *FixedStepAuto* option in SIMULINK R2019b. The training of neural networks was preceded by a data aggregation step. Here, one-minute averages of all measurements were formed [RKF+21].

The neural network training was conducted using Matlab, utilizing the standard back-propagation algorithm with Bayesian regularization, parallelized on 6 CPU cores with 10,000 epochs. The data are randomly divided into three data sets : training (70%), validation (15%) and test (15%).

The inputs used are obtained by principal component analysis based on the singular value decomposition [RKF+21] :

- The stack temperatures $\vartheta_{(1,1,1)}, \vartheta_{(1,2,1)}, \vartheta_{(1,3,1)}$ and the inlet temperatures $\vartheta_{\mathrm{CG,in}}$ and $\vartheta_{\mathrm{AG,in}}$
- The voltage U and electric current I
- The nitrogen and hydrogen mass flows $\dot{m}_{N_2}$, $\dot{m}_{H_2}$ at the anode
- The mass flows $\dot{m}_{\mathrm{CG}}$

Plus the two outputs $\boldsymbol{g}_{\mathbf{Temp1}}, \boldsymbol{g}_{\mathbf{Temp2}}$ mentionned in Section 4.2. They are set to $5K/s$.

The values of the different constants used in the algorithms are also summarized here for reference :

- $T_{n+1} = \lambda T_n$, $\lambda = 0.95$ and $T_0 = 4$ (see Algorithm 5.2)
- $k\max = 50$, $e\max = 1$, $n' = 1$[7.1] (see Algorithm 5.2)
- $\alpha(k) = A\left(\frac{k_{\max} - k}{k_{\max}} + b\right)$, $A = \frac{ew_{\max}}{4}$ and $b = 0.05$ (see Algorithm 5.2)
- $N_{\mathrm{tr}} = 50$ (see Algorithm 5.3)

Due to the low computational efficiency, the results of the algorithm presented in Section 6 will not be presented here. Tests were conducted for $n_{\mathrm{count}} = 10$, but the results are quite random due to the limited number of points to extract a realistic median. This slowness is explained by the quasi-systematic search for a new $\boldsymbol{e_w}$ vector at each iteration. That's why clustering is performed later (see Section 7.3) to determine a finite set of suitable vectors in order to limit the search for $\boldsymbol{e_w}$ vectors to this set, that should greatly increase execution speed.

7.1. A few quick tests were conducted, and the value of n' does not seem to significantly affect the robustness of the found $\boldsymbol{e_w}$ but significantly slows down the code

7.2 Results and analysis

The results presented in Table 7.1 are derived from neural network trainings for which L_1 and L_2 are 15 and 20, respectively. The first row, labeled as GR, corresponds to the case where none of the imposed conditions apply, and only the accuracy of temperature prediction matters. The second row, labeled as GER, represents the case where the e_w vector is set to $(1, 2.5, 7.4, .5, .5, 1)$[7.2]. And the last row GS represents the performance results obtained from the code with simulated annealing. The statistics are based on filtered runs which means :

- The prediction error on temperature must not be a nan value.
- The prediction error on temperature must be less than 10^4

The number of these runs is indicated in the "filtered runs" column. The first column indicates the median error. The second column shows the proportion of simulations that satisfied the conditions on $\boldsymbol{A}$ and $\boldsymbol{b}$ throughout its entire duration. The third column displays the proportion of simulations that met the conditions on $\boldsymbol{A}$ and $\boldsymbol{b}$ during its entire duration and fulfilled the other conditions for 99.9% of the time. The fourth column indicates whether or not simulated annealing was used. The RMS column displays the prediction error returned by the best simulation. Finally, the last column indicates whether this best simulation adheres to the criteria defined for the third column.

	median error	Conditions on $\boldsymbol{A}, \boldsymbol{B}$	All conditions	Simulated Annealing	Filtered runs	Total runs	RMS	All conditions
GR	7.39	0%	0	No	82	102	2.82	No
GER	60.43	40.82%	32.65%	No	98	102	2.82	No
							6.31	Yes
GS	44.89	81.25%	56.25%	Yes	48	51	3.4	Yes

Table 7.1. L_1 and L_2 set to 15 and 20 respectively

In the first row, the median error is much lower than in the other rows. However, none of the conditions necessary for physical interpretation are met. In contrast, in the other two codes, the conditions are met to a certain extent. With the simulated annealing, the criteria on $\boldsymbol{A}$ and $\boldsymbol{b}$ are met more than 4 out of 5 times, and all conditions are met more than half the time. These proportions are twice as high as in the case where the $\boldsymbol{e_w}$ vector was manually determined, demonstrating the utility of simulated annealing not only from a practical perspective but also in terms of performance. Furthermore, although the median error in the first case is much lower, the errors resulting from the best simulation for each of the codes are comparable. The advantage for the simulated annealing is that it achieved a good result with all conditions met.

A more detailed analysis of the code with simulated annealing is presented in Table 7.2. On the left are the statistics regarding the best 15 simulations in terms of RMSE, and on the right are the other simulations. The first column represents the mean of the corresponding variable, the second column represents its standard deviation, and finally, the last column represents the ratio of the standard deviation to the mean of the same variable.

7.2. These values were manually determined by the previous student who worked on this project [Eil23]

	mean	std	std/mean
w_A	3.87	1.16	0.30
w_B	4.76	3.45	0.72
w_{Temp1}	8.22	2.13	0.26
w_{Temp2}	6.61	2.07	0.31
w_{St}	5.94	1.84	0.31
errorRate	0		
perfErrorRate	0.057		

	mean	std	std/mean
w_A	3.93	1.73	0.44
w_B	5.10	3.16	0.62
w_{Temp1}	7.47	2.60	0.35
w_{Temp2}	6.18	2.74	0.44
w_{St}	5.58	2.57	0.46
errorRate	0.087		
perfErrorRate	0.190		

Table 7.2. On the left, statistics for the top 15 simulations in terms of RMSE; on the right, statistics for the other simulations.

It can be observed that the standard deviations in the left table are relatively lower than those on the right. This suggests the presence of a better solution $\boldsymbol{e_w}$ for the system. However, even though the standard deviations are lower, they are still substantial, ranging around 30% of their mean values for the most part, and even reaching up to 72% of the mean value for w_B. This element will likely need to be studied more deeply in order to achieve convergence of $\boldsymbol{e_w}$ (see Section 7.3), which would enable us to reduce the code's execution time.

Additionally, a significant observation emerges. In the left table, the mean of errors are significantly lower than those in the right table. This underscores that for non-zero weights[7.3], the accuracy of predictions increases with physical relevance.

It can also be observed that the weights of the best simulations are not lower than those of the others. The sum of the average values of the components of the ew vectors from Table 7.2 is 29.4 on the left side and 28.26 on the right side, which are very close. This means that the variable v defined in equation (5.2) is not necessary. There is no posterior relationship between prediction accuracy and the amplitude of error weights.

7.3 Values of $\boldsymbol{e_w}$

Given the inability to ascertain a single precise value for $\boldsymbol{e_w}$ using simulated annealing (see Table 7.2), the focus shifts to determining whether a set of promising $\boldsymbol{e_w}$ values exists. To achieve this, the application of a clustering algorithm, the k-means, is intended, followed by the evaluation of results using the silhouette coefficient.

To achieve this, Algorithm 5.3 was executed six times - five times with $N_{\text{tr}}=50$ and once with $N_{\text{tr}}=100$. Subsequently, the $\boldsymbol{e_w}$ values associated with low prediction errors (RMS < 15) were saved. This yielded a total of 69 $\boldsymbol{e_w}$ vectors, which will be subjected to the k-means algorithm. This algorithm aims to identify k clusters, with k being predetermined. We apply the k-means algorithm for 2 to 10 clusters and subsequently analyze these results using the silhouette score.

Figures 7.1, 7.2, and 7.3 display silhouette coefficients with different numbers of clusters. The y-axis indicates the cluster number of a set of points, while the x-axis represents the silhouette coefficient. The red vertical line denotes the mean value of silhouette coefficients. According to the figures below, it is unlikely to have 5, 6 and 7 clusters, as there are clusters in these cases where the silhouette scores are all lower than the mean [PVG+11]. For instance, in the case of 5 clusters, this is observed in cluster number 5. The cases with 8, 9 and 10 clusters are not shown here but are not good either.

7.3. When all the weights are zero except for the prediction weight which has been fixed to 1, this is equivalent to the case GR

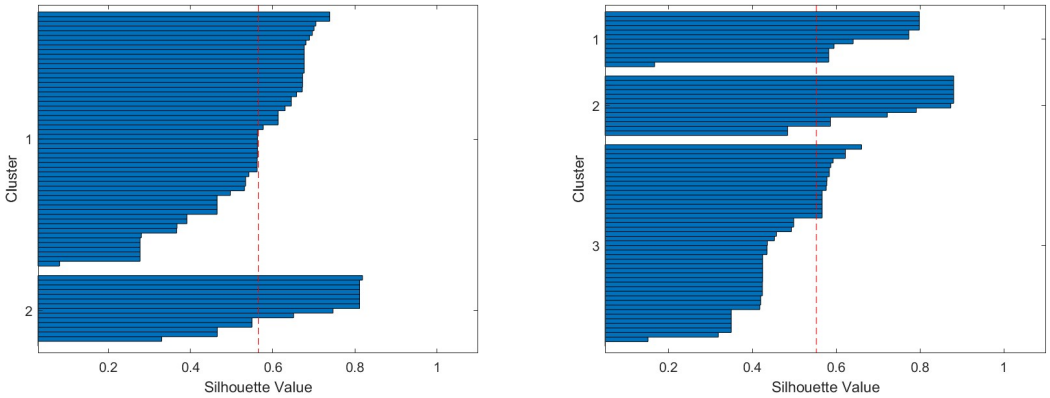


Figure 7.1. (a) Silhouette coefficients with 2 clusters (b) Silhouette coefficients with 3 clusters

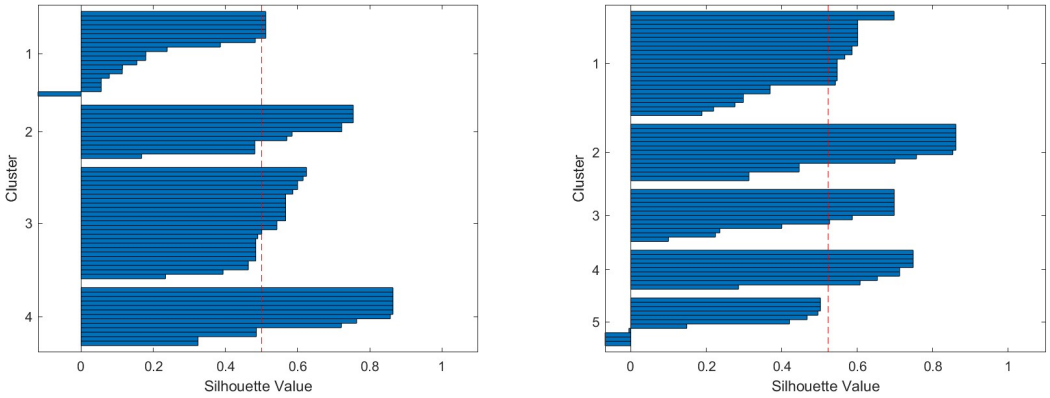


Figure 7.2. (a) Silhouette coefficients with 4 clusters (b) Silhouette coefficients with 5 clusters

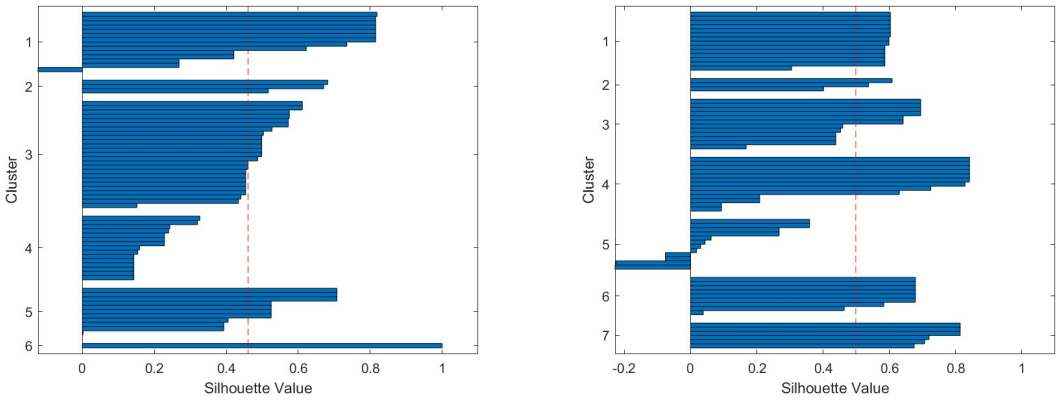


Figure 7.3. (a) Silhouette coefficients with 6 clusters (b) Silhouette coefficients with 7 clusters

As mentionned in Section 5.4 significant instability in clustering, i.e., if the silhouette plot associated with the same total number of clusters changes drastically from one call of k-means to another, then it indicates that the total number of clusters is not suitable for our data. Then in order to analyze the stability of the results, the process was repeated 20 times on the same set of $\boldsymbol{e_w}$ vectors. Table 7.3 presents the number of times good results were obtained

Number of clusters	2	3	4	5	6	7	8	9	10
Good results rate	17	18	12	8	4	3	3	3	2

Table 7.3. Stability of k-means on the set of $\boldsymbol{e_w}$

These results confirm that for a number of clusters k greater than 5, the results are not satisfactory. We can further reject the value 4. Therefore, we are left with a choice between 2 and 3 clusters. Table 7.4 provides statistics on the entire set of 69 $\boldsymbol{e_w}$ vectors. Table 7.5 presents statistics on the clusters with a total of 2 clusters. Table 7.6 offers statistics on the clusters with a total of 3 clusters. According to the results, the clustering does not really distinguish optimal $\boldsymbol{e_w}$ values. The ratio of standard deviation to mean is not significantly lower within the clusters compared to the case described in Table 7.4.

	mean	std	std/mean
w_A	5.02	2.13	0.43
w_B	5.9	3.01	0.51
w_{Temp1}	4.36	3.22	0.74
w_{Temp2}	4.29	2.93	0.68
w_{St}	5.09	2.4	0.47

Table 7.4. Statistics on the entire set of $\boldsymbol{e_w}$

	mean	std	std/mean
w_A	3.71	1.75	0.47
w_B	1.82	1.97	1.08
w_{Temp1}	8.24	2.59	0.31
w_{Temp2}	7.99	1.46	0.18
w_{St}	6.01	2.12	0.35

	mean	std	std/mean
w_A	5.39	2.10	0.39
w_B	7.03	2.14	0.31
w_{Temp1}	3.28	2.46	0.75
w_{Temp2}	3.27	2.35	0.72
w_{St}	4.84	2.43	0.50

Table 7.5. Statistics with two clusters : on the left, statistics for the cluster 1; on the right, statistics for the cluster 2.

	mean	std	std/mean
w_A	4.41	1.67	0.38
w_B	7.24	2.23	0.31
w_{Temp1}	0.51	1.06	2.08
w_{Temp2}	6.78	2.14	0.32
w_{St}	8.54	0.76	0.09

	mean	std	std/mean
w_A	5.55	2.17	0.39
w_B	6.81	2.31	0.34
w_{Temp1}	4	2.15	0.54
w_{Temp2}	2.55	1.74	0.68
w_{St}	4.02	1.86	0.46

	mean	std	std/mean
w_A	3.8	1.85	0.49
w_B	1.56	1.58	1.01
w_{Temp1}	9.12	1.1	0.12
w_{Temp2}	7.89	1.54	0.20
w_{St}	5.53	1.82	0.33

Table 7.6. Statistics with three clusters : on the left, statistics for the cluster 1; on the middle, statistics for the cluster 2; on the right, statistics for the cluster 3.

8 Conclusion

In this study, we have tackled the challenge of optimizing neural networks for modeling complex physical systems, focusing on solid oxide fuel cells. These high-temperature fuel cells hold immense potential in revolutionizing energy conversion, but their complex multiphysics nature has often posed hurdles in achieving accurate and efficient modeling.

Traditional neural networks are often regarded as black-box models due to their opaque internal workings, making it challenging to extract meaningful physical insights, particularly in the context of multiphysics phenomena. However, our approach has taken a distinctive direction, emphasizing the necessity of structured neural networks that adhere to physical constraints. By doing so, we aim to bring about transparency and interpretability to neural network models, enabling us to bridge the gap between predictive power and comprehensible insights.

By structuring the network architecture and incorporating physical constraints, we've aimed to maintain prediction accuracy while maintaining a link to underlying physics. We've explored automated methods to find optimal weight configurations and hidden layer dimensions. This offers computational efficiency and uncovers valid solutions adhering to physical principles.

There are possibilities for improvement, particularly concerning the determination of the optimal number of hidden neurons for layers H and L. One could continue with the previously described approach or explore alternative methods, like the automated application of Principal Component Analysis. This approach has the potential to enhance data understanding and facilitate a more precise selection of relevant features for modeling. These improvements could result in neural networks that are even more efficient and better aligned with the physical constraints of the systems under study.

Bibliography

[BA+17] Marta Boaro, Antonio Salvatore Aricò et al. *Advances in medium and high temperature solid oxide fuel cell technology*, volume 574. Springer, 2017.

[CX01] SH Chan and ZT Xia. Anode micro model of solid oxide fuel cell. *Journal of the Electrochemical Society*, 148(4):0, 2001.

[CYS+18] Paul A Connor, Xiangling Yue, Cristian D Savaniu, Robert Price, Georgios Triantafyllou, Mark Cassidy, Gwilherm Kerherve, David J Payne, Robert C Maher, Lesley F Cohen et al. Tailoring sofc electrode microstructures for improved performance. *Advanced Energy Materials*, 8(23):1800120, 2018.

[Eil23] Sebastian Eilermann. Robust neural network modeling of the thermal behavior of fuel cells under consideration of physical constraints. Master's thesis, Carl von Ossietzky University of Oldenburg, January 2023.

[FRKA20] Wiebke Frenkel, Andreas Rauh, Julia Kersten, and Harald Aschemann. Experiments-based comparison of different power controllers for a solid oxide fuel cell against model imperfections and delay phenomena. *Algorithms*, 13(4):76, 2020.

[Gus07] Bertil Gustafsson. *High order difference methods for time dependent PDE*, volume 38. Springer Science & Business Media, 2007.

[IRK+19] Sara Ifqir, Andreas Rauh, Julia Kersten, Dalil Ichalal, Naima Ait-Oufroukh, and Said Mammar. Interval observer-based controller design for systems with state constraints: application to solid oxide fuel cells stacks. In *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 372–377. IEEE, 2019.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RKA18] Andreas Rauh, Julia Kersten, and Harald Aschemann. An interval observer approach for the online temperature estimation in solid oxide fuel cell stacks. In *2018 European Control Conference (ECC)*, pages 1596–1601. IEEE, 2018.

[RKF+21] Andreas Rauh, Julia Kersten, Wiebke Frenkel, Niklas Kruse, and Tom Schmidt. Physically motivated structuring and optimization of neural networks for multi-physics modelling of solid oxide fuel cells. *Mathematical and Computer Modelling of Dynamical Systems*, 27(1):586–614, 2021.

[RSA15a] Andreas Rauh, Luise Senkel, and Harald Aschemann. Interval-based sliding mode control design for solid oxide fuel cells with state and actuator constraints. *IEEE Transactions on Industrial Electronics*, 62(8):5208–5217, 2015.

[RSA15b] Andreas Rauh, Luise Senkel, and Harald Aschemann. Reliable sliding mode approaches for the temperature control of solid oxide fuel cells with input and input rate constraints. *IFAC-PapersOnLine*, 48(11):390–395, 2015.

[RSKA16] Andreas Rauh, Luise Senkel, Julia Kersten, and Harald Aschemann. Reliable control of high-temperature fuel cell systems using interval-based sliding mode techniques. *IMA Journal of Mathematical Control and Information*, 33(2):457–484, 2016.

[Spi18] Luca Spiridigliozzi. *Doped-Ceria Electrolytes: Synthesis, Sintering and Characterization*. Springer, 2018.

[TMK+16] Maciej Tatko, Michał Mosiałek, Aneta Kędra, Elżbieta Bielańska, Małgorzata Ruggiero-Mikołajczyk, and Paweł Nowak. Thermal shock resistant composite cathode material sm 0.5 sr 0.5 coo 3–δ–la 0.6 sr 0.4 feo 3–δ for solid oxide fuel cells. *Journal of Solid State Electrochemistry*, 20:143–151, 2016.

Appendix A Assessment report

ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Carl von Ossietzky University of Oldenburg

Adresse / *Address* Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany

Tél / *Phone (including country and area code)* +49-(0)441-798-4195

Nom du superviseur / *Name of internship supervisor*
Prof. Dr.-Ing. RAUH Andreas
Fonction / *Function* Head of the research group "Distributed Control in Interconnected Systems (DCIS)" and Dean of the School for Computer Science, Business Administration, Economics, and Law

Adresse e-mail / *E-mail address* andreas.rauh@uni-oldenburg.de

Nom du stagiaire accueilli / *Name of intern* REN Kevin

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? *Was the initial contract carried out to your satisfaction?* (A) B C D E F
- Manquait-il au stagiaire des connaissances ? *Was the intern lacking skills?* ☐ oui/*yes* ☒ non/*no*

  Si oui, lesquelles ? / *If so, which skills?* /

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)* (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* The student did not obtain a visa for Germany. Hence, supervision took place online. Communication took place ~~on~~ on a regular basis in a well-prepared manner.

7

Version du 05/04/2019

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

A B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* The student has been responsive, well-prepared during online meetings and interested in learning new approaches.

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? A B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? A B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : A B C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☒ oui/*yes* ☐ non/*no*

Fait à Oldenburg, le 12/09/2023
In _____, on _____

Signature Entreprise Andreas Rauh
Company stamp

Signature stagiaire
Intern's signature

Carl von Ossietzky Universität Oldenburg
Fakultät II
Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik
Abt. Verteilte Regelung in vernetzten Systemen
Prof. Dr. Andreas Rauh
D-26111 Oldenburg

Merci pour votre coopération
We thank you very much for your cooperation

8

Version du 05/04/2019