Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. xxx (xxxx) xxx
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Highlights

Prediction and identification of physical systems by means of Physically-Guided Neural Networks with meaningful internal layers

Comput. Methods Appl. Mech. Engrg. xxx (xxxx) xxx

Jacobo Ayensa-Jiménez, Mohamed H. Doweidar, Jose A. Sanz-Herrera, Manuel Doblaré*

- Physically informed Neural Networks are used to get physical predictions.
- Only measurable values are used to train the network.
- The NN topology permits to identify hidden variables with internal neuron values.
- Both prediction and characterization problems are solved.
- This method accelerates training, and reduces the data required for similar accuracy.
- It filters partly the noise in the data and provides improved extrapolation capacity.

Please cite this article as: J. Ayensa-Jiménez, M.H. Doweidar, J.A. Sanz-Herrera et al., Prediction and identification of physical systems by means of Physically-Guided Neural Networks with meaningful internal layers, Computer Methods in Applied Mechanics and Engineering (2021) 113816, https://doi.org/10.1016/j.cma.2021.113816.

CMA: 113816

ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. xxx (xxxx) xxx

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Prediction and identification of physical systems by means of Physically-Guided Neural Networks with meaningful internal layers

Jacobo Ayensa-Jiménez[a], Mohamed H. Doweidar[b], Jose A. Sanz-Herrera[c], Manuel Doblaré[d,*]

[a] *Mechanical Engineering Department, Aragón Institute of Engineering Research (I3A), University of Zaragoza, Mariano Esquillor, S/N, 50018 Zaragoza, Spain*

[b] *Mechanical Engineering Department, School of Engineering and Architecture (EINA), University of Zaragoza, María de Luna, S/N, 50018 Zaragoza, Spain*

[c] *Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Camino de los Descubrimientos, S/N, 41092 Sevilla, Spain*

[d] *Aragón Institute of Engineering Research (I3A), University of Zaragoza, Centro de Investigación Biomédica en Red en Bioingeniería, Biomateriales y Nanomedicina (CIBER-BBN), Instituto de Investigación Sanitaria, Mariano Esquillor, S/N, 50018 Zaragoza, Spain*

Received 23 September 2020; received in revised form 5 March 2021; accepted 18 March 2021
Available online xxxx

Abstract

Substitution of well-grounded theoretical models by data-driven predictions is not as simple in engineering and sciences as it is in social and economic fields. Scientific problems suffer many times from paucity of data, while they may involve a large number of variables and parameters that interact in complex and non-stationary ways, obeying certain physical laws. Moreover, a physically-based model is not only useful for making predictions, but to gain knowledge by the interpretation of its structure, parameters, and mathematical properties. The solution to these shortcomings seems to be the seamless blending of the tremendous predictive power of the data-driven approach with the scientific consistency and interpretability of physically-based models.

We use here the concept of Physically-Guided Neural Networks (PGNN) to predict the input–output relation in a physical system, while, at the same time, fulfilling the physical constraints. With this goal, the internal hidden state variables of the system are associated with a set of internal neuron layers, whose values are constrained by known physical relations, as well as any additional knowledge on the system. Furthermore, when having enough data, it is possible to infer knowledge about the internal structure of the system and, if parameterized, to predict the state parameters for a particular input–output relation. We show that this approach, besides getting physically-based predictions, accelerates the training process, reduces the amount of data required to get similar accuracy, partly filters the intrinsic noise in the experimental data and improves its extrapolation capacity.

Keywords: Physically Guided Neural Networks; Explanatory artificial intelligence; Data-driven simulation-based engineering and sciences; Model identification; Parameter identification; NN prediction improvement

* Corresponding author.
E-mail address: mdoblare@unizar.es (M. Doblaré).

https://doi.org/10.1016/j.cma.2021.113816

1. Introduction

Science has progressed historically through the fruitful interaction between theory and experiments, or better, between hypotheses and data. Additionally, since the appearance of computers in the fifties, and accelerated in the nineties of the XXst century, simulation has been progressively recognized as the third pillar of the scientific method [1].

To describe a physical phenomenon, one states the mathematical equations that control the evolution of a set of variables (position, momentum, temperature, entropy, etc.) that completely determine the state of the system. That evolution depends upon a set of external stimuli, that are assumed to be known, and upon the state itself. In this context, we distinguish between two kinds of equations: universal physical principles (conservation laws and physical inequalities), and the internal state equations that compile the averaged behavior of the system from its particular internal structure. The ability of any physical–mathematical model to accurately represent the reality is directly related to the quality of the simplification hypotheses that drive to those state equations and to the available experimental data required to identify the associated parameters.

This combination of universal physical principles and phenomenological state models under well-contrasted hypotheses has demonstrated to be highly effective to accurately predict the state and evolution of big and complex realistic problems, while keeping them mathematically tractable.

A new paradigm is raising, however, based on our increasing ability to collect, store, analyze, and extract information from high volumes of data, a capability that is accelerating at an unprecedented rate [2,3]. Based on the success of Data Science and Artificial Intelligence in fields like e-commerce [4], social sciences [5], healthcare [6], language recognition [7], image-based predictions [8], etc., they are also gaining prominence in simulation-based engineering and sciences (SBES).

However, data gathering in Physics is soaked by centuries of scientific knowledge and the associated human bias [9–12]; so, a "blind" algorithm without any information on that bias may lead to wrong predictions. Also, scientific problems suffer many times from paucity of data while involving a large number of variables that interact in complex and non-stationary ways. Therefore, we can expect poor predictive capability of purely data-based approaches in problems far from the training set. Finally, a physically-based model is not only useful for making predictions, but it is expected to help in gaining knowledge by the interpretation of its structure, parameters, and mathematical properties. In fact, physical interpretability is, in many cases, at least as important as predictive performance. It is not strange therefore the important efforts made in "whitening" the "black box" way of working of current machine-learning predictive algorithms [13].

One possible solution to this shortcoming of data-only models is the seamless blending of their tremendous predictive power with the scientific consistency and interpretability of physically-based models. The term coined for this hybrid paradigm is physically-guided data science (PGDS) [14–17]. A straightforward application of PGDS techniques is dynamic data-driven systems (DDSBES) [18**?**–23]. However, one of the most important drawbacks in current data-driven approaches is the need of explicitly defining the cloud of experimental values that identifies the internal state model (e.g. material constitutive equations in solid mechanics) with a sufficient number of points in the whole range of interest. This forces us to perform extensive experimental campaigns that are costly in time and money and whose results rely on strong assumptions on the experimental model itself (e.g. uniform distribution of stresses in uniaxial tests), that cannot be overcome due to the non-observable (non-measurable) character of some of such state variables (e.g. stresses).

An opposite perspective is integrating physical knowledge into data science models, that is, to constrain the prediction domain of the standard data model by physical constraints. Of course, this approach may be extended to any known relation between the input–output variables. One simple example of this addition of physical knowledge was made in [14] to fill incomplete data sets coming from experimental campaigns in a reliable way. Another more powerful approach is using physical knowledge to inform and improve the data prediction capability of neural networks [15–17,24–29]. However, in all these works, the physical information was introduced directly as relations between the input and output layers. Only in [16,17] a first attempt was made to provide the network with some explanatory capacity by adding some of the parameters associated with the internal state model as output.

In this paper, we extend such explanatory capability by establishing a general approach in which we distinguish between the universal physical laws and the internal state equations. The former are treated as constraints imposed by the particular Physics between neuron values in the NN, with an appropriate topology, while the latter are derived as a direct NN outcome. This will permit us to identify some of the internal neurons with internal (in

general non-observable) state variables. The objective of this work is therefore to introduce this new methodology of *Physically Guided Neural Networks with Internal Variables* (PGNNIV) to predict both the input–output relation in a physical system from a sufficient set of data, as well as to infer knowledge on the system internal structure, but always considering the constraints imposed by Physics. The corresponding NN is trained by only observable data (e.g. displacements or velocities, forces, etc.), extracting the internal non-observable values (e.g. stresses) from the NN output.

We show that this methodology shows a better performance in terms of faster convergence, less need of data, data noise filtering and bias correction, and extrapolation capacity.

2. Physically-Guided Neural Networks with Internal Variables. Concept, formulation and types of applications

2.1. General framework

We identify immediately two types of state variables in any averaged phenomenological theory: (i) observable (measurable) variables that can be local such as the position, pressure or temperature or integral as energy variations. These will be denoted as $\boldsymbol{u}$, and collect all the **essential variables** of the problem, that is, the minimum set of independent, in general spatial and time-dependent variables that define the observable state of the system; (ii) **internal variables**, not always directly measurable, that are model-specific. In general, these internal state variables collect the changes in the internal structure of the system. They will be denoted as $\boldsymbol{\eta}$.

In the same way, we stated already that there are two types of equations: (i) **universal physical laws**, valid for any problem in a certain context (e.g. non-relativistic mechanics), such as conservation of mass, linear and angular momenta and energy. They define the time evolution of the system such as $\dot{\boldsymbol{u}} = \boldsymbol{G}(\boldsymbol{u}, \boldsymbol{\eta}, \boldsymbol{f})$ ($\boldsymbol{G}(\boldsymbol{u}, \boldsymbol{\eta}, \boldsymbol{f}) = \boldsymbol{0}$ in the non-transient case), with $\boldsymbol{G}$ a set of functions, universal for a particular family of problems and $\boldsymbol{f}$ the external stimuli assumed to be known; (ii) **state equations** that define the averaged evolution of the internal variables in terms of the current state of the system ($\boldsymbol{u}$ and $\boldsymbol{\eta}$) and a set of internal parameters $\boldsymbol{\lambda}$, $\dot{\boldsymbol{\eta}} = \boldsymbol{H}(\boldsymbol{\eta}, \boldsymbol{u}, \boldsymbol{\lambda})$ ($\boldsymbol{\eta} = \boldsymbol{H}(\boldsymbol{u}, \boldsymbol{\lambda})$ in the non-transient case, redefining $\boldsymbol{H}$ if necessary). These equations are most times phenomenological. Thus, their functional form and associated parameters have to be determined from reasonable assumptions and experimental tests. These experimental needs are one of the main bottlenecks to set up a model representing a physical system.

2.2. Physically-Guided Neural Networks with Internal Variables

2.2.1. Mathematical formulation

To predict the value of the essential variables, we define the architecture of our PGNNIV according to the following recipe:

1. We identify the output layer with (some or all) the values of the essential variables, $\boldsymbol{u}$, or some directly related quantities.
2. The input layer corresponds to known values such as the external stimuli $\boldsymbol{f}$ or boundary conditions. We can also interchange the role of the input and output variables.
3. Some predefined internal layers (PILs) are associated with the internal state variables $\boldsymbol{\eta}$. The values of such neurons may be recovered after convergence, getting the values of the internal state variables from the solution of the system for a particular input. The difference between PILs and common internal layers is that the mathematical constraints are applied to the neurons of the former, *guiding* the learning process.
4. The rest of the internal layers, connecting the input, output and PILs, follows the standard approach in NN, so they are able to "discover" the complex relations hidden in the function $\boldsymbol{H}$.
5. Finally, the universal laws stated in $\boldsymbol{G}$ are established in the NN as constraints between input, output and internal layers.

Any additional physical knowledge of the system can be additionally imposed in a similar way. Moreover, we can further supply partial or total information about the internal state model by defining a parametric state equation $\boldsymbol{\eta} = \boldsymbol{H}(\boldsymbol{u}; \boldsymbol{\lambda})$, by establishing the appropriate topology for the state model network and adding additional constraints.

As a result of the learning process, the relationship $\boldsymbol{\eta} = \boldsymbol{H}(\boldsymbol{u})$ is learned and so it is the input–output relationship. Therefore, Physically-Guided Neural Networks with Internal Variables have both predictive and explanatory capacity.

The state model is then characterized by a neural network topology and parameters $\boldsymbol{\eta} = \mathsf{H}(\boldsymbol{u}) = \boldsymbol{H}(\boldsymbol{u}; \boldsymbol{W})$, instead of an explicit relationship, where $\boldsymbol{W}$ represents the weights and biases of the neural network that are obtained during the training process. Sans serif notation for a functional dependence ($\boldsymbol{y} = \mathsf{Y}(\boldsymbol{x})$, $\boldsymbol{\eta} = \mathsf{H}(\boldsymbol{u})$) is here used to represent an implicit input–output relation in a NN. Indeed, the universal approximation theorem guarantees that a sufficiently regular function may be approximated by a specific neural network with a sufficient number of layers and neurons and convenient activation functions [30–33], so this second approach is, at least, as general as the standard one, also unfolding the benefits of the neural network hardware (fast computation with GPU and TPU, cloud and distributed computing, etc.) and software such as Keras and TensorFlow (modularity, pluggability, fast generalization capability, etc.). All this allows for high-performance computing capabilities and scalability [34], for major model flexibility that allows capturing strong non-linearities [35,36] and for soundness with respect to statistical data (heteroskedasticity, non-normality, etc.) [37–39].

Remark 1. We have to remark that these variables may be spatial and/or time fields depending on the location $\boldsymbol{x}$ and/or time t. In these cases, we shall consider that a previous discretization step has been applied, so the time-position independent interpolating variables are those of the associated discretized problem. Therefore, the same approach can be used both for non-transient problems or for transient ones, using as variables $\boldsymbol{u}$ the algebraic values that define the approximated field at a certain time and interpolation point, following a step-by-step continuation approach.

Remark 2. If no constraint is applied and no PIL is defined, we recover the classical Neural Network framework. If the constraints are applied to the input or output layers, we recover the formulation developed by other authors for Physically-Guided Neural Networks [15,16]. A similar approach to the one here presented was also addressed in [16] for partial differential equations, but without using the PIL concept that is original, up to the authors' knowledge.

Remark 3. This framework allows the scientist to work only with directly measurable variables and fields, without the need of establishing any *a priori* assumption on the expression of the internal variables, which is fundamental, since internal (non-measurable) variables are, indeed, mathematical constructs that are now determined as a byproduct of the predictive problem.

2.2.2. Construction of Physically-Guided Neural Networks with Internal Variables

Denoting the input variable $\boldsymbol{x}$ as $\boldsymbol{y}_0$, each hidden layer of n_i neurons, $\boldsymbol{y}_i$, $i = 1, \dots, L$ is defined by a functional relation:

$$\boldsymbol{y}_i = \phi(\boldsymbol{y}_{i-1}\boldsymbol{W}_i + \boldsymbol{b}_i), \tag{1}$$

where $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$, $i = 1, \dots, L$ are the weights and biases, the parameters of the model, and $\phi : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is an activation function. With this notation, the output variable is $\boldsymbol{y} = \boldsymbol{y}_L$. The network is symbolically represented by the relationship $\boldsymbol{y} = \mathsf{Y}(\boldsymbol{x})$ or, denoting by $\boldsymbol{W}$ the whole set of weights and biases for a given network topology, $\boldsymbol{y} = \boldsymbol{Y}(\boldsymbol{x}; \boldsymbol{W})$. Given a set of *ground truth* data points $\mathcal{D} = \{(\bar{\boldsymbol{x}}^i, \bar{\boldsymbol{y}}^i) | i = 1, \dots, N\}$, a quadratic mean error is used to evaluate the network performance, $\mathrm{MSE}(\boldsymbol{W}|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \left\| \bar{\boldsymbol{y}}^i - \boldsymbol{Y}(\bar{\boldsymbol{x}}^i; \boldsymbol{W}) \right\|^2$.

Now, it is possible to train the neural network by minimizing the function MSE, getting the optimal set of weights and biases $\boldsymbol{W}$. Let us now impose some constraints such that some neuron values satisfy some (physically-based) equations, including universal laws, manifold constraints and boundary conditions. Without loss of generality, we denote all these functions by $\boldsymbol{R}_j$, $j = 1, \dots, r$.

$$\boldsymbol{R}_j(\boldsymbol{y}_0, \dots, \boldsymbol{y}_L) = \boldsymbol{0}, \quad j = 1, \dots, r. \tag{2}$$

We can reformulate the minimization problem as:

$$\min_{\boldsymbol{W}} \quad \mathrm{MSE}(\boldsymbol{W}|\mathcal{D}) \qquad \text{s.t.} \qquad \boldsymbol{R}_j(\boldsymbol{W}|\mathcal{D}) = \boldsymbol{0}, \quad j = 1, \dots, r. \tag{3}$$

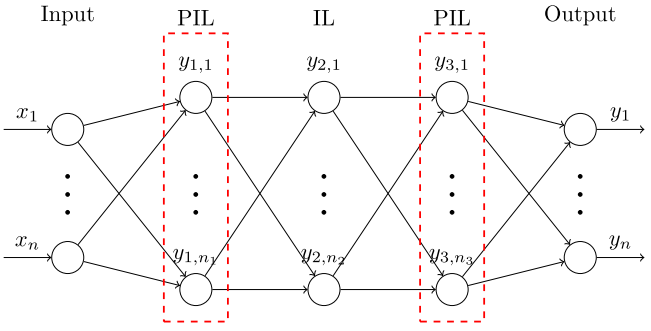


Fig. 1. Physically-Guided Neural Network for a three hidden-layered network. The red dashed rectangles indicate the neurons in which a certain constraint is applied (PILs). The number of internal layers between the input layer and layer 1, layer 1 and layer 3 and between layer 3 and the output layer can be increased to allow more complex models. It is the physical constraint the one that provides the PILs 1 and 3 a physical interpretation as state variables.

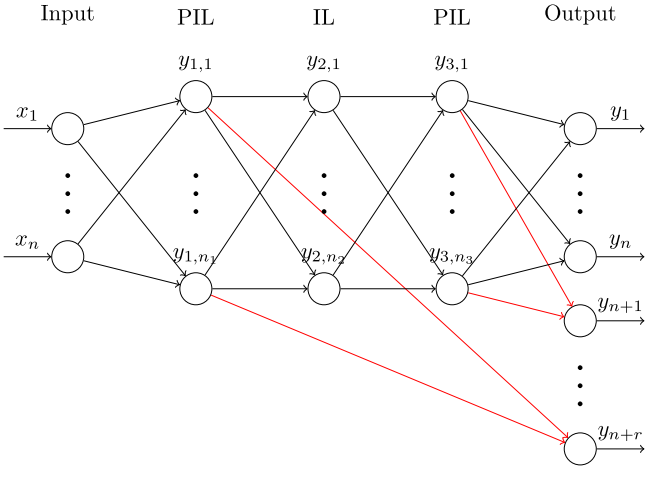


Fig. 2. Augmented Neural Network equivalent to the Physically-Guided Neural Network. Each constraint is replaced by an extra output representing the value of the constraint that, ideally, should be null.

If some of the constraints are expressed by an explicit equation, we can modify the problem (3) to make this constraint disappear from the general formulation. Fig. 1 illustrates a Physically-Guided Neural Network for a three hidden-layered neural network.

It is possible to reformulate (3) using a penalty approach and defining r penalty parameters p_j:

$$\min_{\boldsymbol{W}} \quad \text{MSE}(\boldsymbol{W}|\mathcal{D}) + \sum_{j=1}^{r} p_j \left\| \boldsymbol{R}_j(\boldsymbol{W}|\mathcal{D}) \right\|^2 . \tag{4}$$

This approach allows the implementation of the problem in a standard Neural-Network framework (i.e. TensorFlow@Python) by just defining an adapted loss function that includes the penalty term OF = MSE + PEN with $\text{PEN}(\boldsymbol{W}) = \sum_{j=1}^{r} p_j \left\| \boldsymbol{R}_j(\boldsymbol{W}|\mathcal{D}) \right\|^2$.

Note that (4) may be interpreted as an auxiliary neural network with input $\boldsymbol{x}$ and output $\hat{\boldsymbol{y}} = (\boldsymbol{y}; \boldsymbol{y}_P)$, being $\boldsymbol{y}_P$ a new set of output variables, $\boldsymbol{y}_P = (\boldsymbol{R}_1, \ldots, \boldsymbol{R}_r)$ with physical meaning, whose *ground-truth* value is always 0, that is $\boldsymbol{R}_j = \boldsymbol{0}$ as illustrated in Fig. 2.

The inclusion of inequalities in the presented framework is possible using the ReLU function. Indeed, the inclusion of a term in the penalty function with the structure $p\text{ReLU}(f(\boldsymbol{W}, \mathcal{D}))$ guarantees that if p is high enough, $\text{ReLU}(f(\boldsymbol{W}, \mathcal{D}))$ has to be the smallest possible, ensuring that $\text{ReLU}(f(\boldsymbol{W}, \mathcal{D})) \to 0$ and therefore $f(\boldsymbol{W}, \mathcal{D}) \leq 0$.

This approach has two main advantages that match the two spearheads against Artificial Neural Networks (ANN) methods:

1. From a physical point of view, we postulate some extra conditions onto the hidden variables, which allow us to interpret them as true physically-based features, that is, as state variables of the physical problem, overcoming the *black-box* problem of neural networks [40–42].
2. As the search space is reduced via constraints, the optimization algorithm is expected to learn faster, with less information, to filter the noise incompatible with the problem physics and to discard solutions without physical sense.

2.3. Types of problems in which Physically-Guided Neural Networks with Internal Variables may be applied

We can think of several families of problems, namely:

1. **Prediction problems:** The goal is now to predict the value of a set of dependent output variables $\boldsymbol{y}$ from other independent measurable ones $\boldsymbol{x}$. The material constitutive model or state equation is assumed to be frozen and therefore there exists an (unknown) relationship, $\boldsymbol{H}$, whose functional form or properties have to be revealed. Consequently, the state equation is context-dependent and may be formulated as $\boldsymbol{\eta} = \boldsymbol{H}(\boldsymbol{y})$ (or $\boldsymbol{\eta} = \boldsymbol{H}(\boldsymbol{x})$).
   When solving prediction problems, different objectives may be followed:
   (a) **Pure predictive problem**. We establish a direct correspondence between observable variables for a fixed system (same geometry and internal structure), without any constraint nor explicit establishment of PILs. We use the NN in the standard black-box manner to get the correlation between the input and output variables to predict, after training, the latter for a particular input, without any knowledge of the physical system This has been done frequently in the last decades [43–45].
   (b) **Pure predictive problem with input–output constraints**. The only difference with the above is the assumption (or knowledge) of some input–output relations. Those relations are imposed via external constraints onto the objective function of the NN without using the concept of internal state variables or PILs. Some particular problems have been solved very recently using this methodology [16,24].
   (c) **Predictive and explanatory problems with internal hidden variables and constraints**. We add, instead, a PIL with the physical meaning of internal state non-observable variables and include, explicitly, the physical laws between them and the directly observable external stimuli and the boundary conditions by constraints in the NN, thus helping the NN to "know" that the system should fulfill such conservation laws. The unknown constitutive model is then obtained as an implicit relation between such internal state variables and the input ones.
   (d) **Predictive and explanatory problems to identify fixed internal parameters**. We add to the previous problem additional information on the structure of the constitutive model via new constraints between PILs, adding (or not) information on the values or ranges of the constitutive parameters. It is therefore possible to ask the NN to predict the particular values of the parameters for a certain predefined explicit constitutive model structure. The model has to be postulated (partially or totally) *a priori* and the searched parameters are obtained as output, reaching some explanatory capacity. This approach has already been explored by some authors without the use of PILs, which limits it to using only measurable variables [16]. As a particular case, the model may be perfectly defined, so the network has no explanatory capacity and the methodology presented may be seen as a pure dimensionality reduction technique or an offline calculator (response surface).
   (e) **Model selection**. The idea is to test a set of potential constitutive models as in the previous case, getting the optimal parameters for each of them and then identifying the most likely as such with the lowest loss function. The one finally selected will be that showing the best performance in terms of the error function. As commented, a specific case is when $\boldsymbol{H}$ is totally specified, that is, $\boldsymbol{\lambda}$ are not learned from the data.

   These two latter examples are similar to classical model regression, but using NN software, hardware and methods, with their advantages in terms of computational cost and distributed computation.

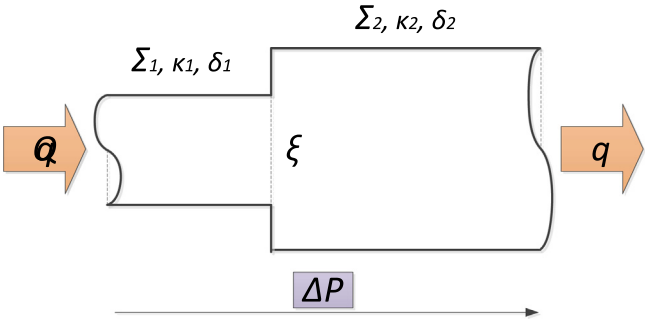


Fig. 3. Scheme of the illustrative example. Hydraulic head loss along a piecewise cross-section pipe.

2. **Characterization problems**: The goal here is to characterize the parameters of a pre-established constitutive model or state equation for different macroscopic materials. The material constitutive model or parameters are assumed to vary from one training data to another. Therefore, for the problem to make sense, enough physical information about the material response must be provided. The inputs are, consequently, the stimulus and the response of the material, $\boldsymbol{x} = (\boldsymbol{u}, \boldsymbol{f})$. The output is composed of variables related to the state model. Two classes of descriptors may be provided as output variables.

   - **Structural descriptors:** Any functional descriptor of the model characteristics (e.g. spatial homogeneity or time invariance, anisotropy or symmetries, linearity, memoryless, damage accumulation, etc.).
   - **Prescribed model parameters:** parameters appearing in the mathematical expression of a given empirical equation. For instance, the Young modulus E and the Poisson ratio ν.

In this second family, we can distinguish the same types of problems as in the predictive one, with analog treatment of the different involved variables. The difference is that the particular state model, defined by an appropriate subset of the complete NN or by a set of parameters associated with an *a priori* defined parametric model, is part of the output variables.

These types of problems may be combined in mixed ones. For example, we can include a PIL associated to gradients of the essential variable (e.g. strains in Mechanics) with the corresponding defining constraint with the input/output layer (displacements) and another one associated to internal hidden variables (stresses) constrained by the conservation law (conservation of linear momentum). These two layers are connected by a subset of the whole NN that identifies the state model defined between both (stress–strain constitutive model).

3. Examples of application

Next, we consider several examples to illustrate the methodology and the different types of applications. We present the two approaches described: prediction and characterization. The fundamental results are shown, while different aspects about the method performance will be addressed in the Discussion section.

3.1. Problem statement

Let us consider a pipe segment of length δ_1, with a circular cross-section of diameter D_1, and a sudden change in its circular cross-section to another segment of length δ_2, with the same shape but with a bigger diameter D_2 (Fig. 3). The initial objective is to compute the head pressure loss, Δp, along the length of the whole pipe in the steady-state regime, assuming fluid incompressibility. Using the Bernoulli equation, along a streamline, the hydraulic head is defined as $h = \frac{v^2}{2g} + z + \frac{p}{\gamma}$, with $\gamma = \rho g$, being ρ the density of the fluid, g the gravity acceleration, z the elevation, so that Δh is the hydraulic head loss due to eddy dissipation and wall friction that corresponds to the state equation of the problem, so it has to be characterized by means of: (i) additional assumptions, usually with poor accuracy; (ii) experimental tests with the corresponding fitted phenomenological equations; (iii) simulations with complex fluid flow models. Here, we shall assume that the hydraulic head loss is associated with two physical phenomena: (i) viscous dissipation distributed along the pipe and (ii) localized dissipation at the pipe expansion.

For the first physical phenomenon, we have distributed losses along a streamline, so assuming the flow as unidimensional and considering a horizontal pipe, we can write:

$$\frac{d}{dx}\left(\frac{v^2}{2g}+\frac{p}{\gamma}\right)=\frac{dh}{dx}=-i, \tag{5}$$

with i the hydraulic head slope.

It is common to express the hydraulic head slope in terms of the fluid velocity by using the Darcy–Weisbach expression [46]:

$$i=f_D\frac{1}{2g}\frac{v^2}{\Phi}, \tag{6}$$

where Φ is the hydraulic diameter of the pipe and f_D the Darcy friction factor, an empirical coefficient that, again, is determined by additional hypotheses or semi-empirical expressions. Here, for illustrative purposes, we adopt a common empirical model: the Hazen–Williams expression for the hydraulic head slope that considers the fluid viscosity and the pipe roughness simultaneously [47]:

$$i=\lambda\left(\frac{q}{\kappa}\right)^{\alpha}\Phi^{\beta}, \tag{7}$$

with $q=\Sigma v$ the flow rate (Σ is the cross-section area), $\lambda=10.67$, $\alpha=1.8520$, $\beta=-4.8704$, κ the roughness parameter of the pipe wall and Φ its hydraulic diameter.

If the pipe has the same properties along a certain distance δ, the previous expression can be immediately integrated, getting:

$$\Delta h=\frac{dh}{dx}\delta=-i\delta=-\lambda\left(\frac{q}{\kappa}\right)^{\alpha}\Phi^{\beta}\delta. \tag{8}$$

If we now move to the second term of the energy loss, due to the sudden pipe expansion, a typical approach is the so-called Borda–Carnot equation [48]:

$$\Delta h=\xi\frac{1}{2g}\left(1-\frac{\Sigma_1}{\Sigma_2}\right)^2 v_1^2, \tag{9}$$

where Σ_1 and v_1 are the cross-section area and flow velocity before the expansion and Σ_2 the cross-section area after expansion. ξ is, again, an empirical coefficient accounting for the magnitude of the viscous eddy dissipation.

Assuming an almost uniform flow velocity profile, which is the case for fully developed flows, it is possible to derive from the mass and momentum conservation equation that $\xi\simeq 1$ [48].

Combining these two hydraulic head loss phenomena, we finally write:

$$(\Delta h)_1=\lambda\left(\frac{q}{\kappa_1}\right)^{\alpha}\Phi_1^{\beta}\delta_1, \tag{10a}$$

$$(\Delta h)_e=\xi\frac{1}{2g}\left(1-\frac{\Sigma_1}{\Sigma_2}\right)^2 v_1^2, \tag{10b}$$

$$(\Delta h)_2=\lambda\left(\frac{q}{\kappa_2}\right)^{\alpha}\Phi_2^{\beta}\delta_2, \tag{10c}$$

In terms of the pressure drop:

$$(\Delta p)_1=\lambda\gamma\left(\frac{q}{\kappa_1}\right)^{\alpha}\Phi_1^{\beta}\delta_1, \tag{11a}$$

$$(\Delta p)_e=\frac{1}{2}\rho q^2\left[\left(\frac{1}{\Sigma_2^2}-\frac{1}{\Sigma_1^2}\right)+\xi\left(\frac{1}{\Sigma_1}-\frac{1}{\Sigma_2}\right)^2\right], \tag{11b}$$

$$(\Delta p)_2=\lambda\gamma\left(\frac{q}{\kappa_2}\right)^{\alpha}\Phi_2^{\beta}\delta_2. \tag{11c}$$

Eq. (11) describes the whole physics of the model when Hazen–Williams and Borda–Carnot loss models are assumed. Besides the empirical equation relating the energy losses with the velocity, there is another underlying

universal physics inherent to the problem, although part of it has been already used even if it has been masked: the momentum conservation equation has been used in the derivation of the Bernoulli equation while the energy conservation has been applied to get Eq (8). Another universal equation is mass conservation (constant flow equation), which was used only in Eq. (11b) but will be used again later.

Let us now use the two different approaches, prediction or characterization, to solve this simple problem.

- **Prediction problem:**
  The aim here is to predict a pressure drop Δp from a given flow q through the pipe. This problem is illustrative in the sense that we know a conserved quantity of the problem (the mass) and we want to know the functional dependence between another physical variable (pressure drop) and this conserved quantity (equivalent to velocity). Taking into account that for circular-based cylindrical pipes $\Phi_i = \sqrt{\frac{4\Sigma_i}{\pi}}$, and renaming the parameters, we can write Eq. (11) as:

$$\Delta p = \lambda_1 q^2 + \lambda_2 q^{\lambda_3}, \tag{12}$$

where:

$$\lambda_1 = \frac{1}{2}\rho\left[\left(\frac{1}{\Sigma_2^2} - \frac{1}{\Sigma_1^2}\right) + \xi\left(\frac{1}{\Sigma_1} - \frac{1}{\Sigma_2}\right)^2\right], \tag{13a}$$

$$\lambda_2 = \frac{2\lambda\gamma}{\sqrt{\pi}} \sum_{i=1}^{2} \frac{\Sigma_i^{\beta/2}}{\kappa_i^{\alpha}}\delta_i, \tag{13b}$$

$$\lambda_3 = \alpha. \tag{13c}$$

  This model may be seen as a physically-based mathematical relation, relating one input, $x = q$ to an output, $y = \Delta p$ variable, by a function $y = Y(x; \boldsymbol{\lambda})$, that includes three formal parameters, λ_1, λ_2 and λ_3, easily obtained from:

  - The cross-section areas of the pipe: Σ_i, $i = 1, 2$.
  - The roughness of the pipe wall: κ_i, $i = 1, 2$.
  - The lengths of different sections of the pipe: δ_i, $i = 1, 2$.
  - Some physical parameters: density ρ and gravitational acceleration g.
  - The model parameters: the exponents $\alpha = 1.8520$ and $\beta = -4.8704$ and the coefficient $\lambda = 10.67$. The values are taken in I.S. units.

  Note that this model is highly nonlinear and, despite its conceptual simplicity, it is not that easy to solve using data-based approaches.

  Depending on the selected approach:

  1. We try to learn the relationship $\Delta p = \mathsf{Y}(q)$.
  2. We learn Δp from the velocities, $\Delta p = Y(v_0, v_1, v_2)$, where $v_i = \frac{q}{\Sigma_i}$. This is a simple but illustrative example of defining a new state variable from the input variable q.
  3. We learn Δp from the flow q but by means of local pressure gradients, $\Delta p = Y(w_1, w_2) = \delta_1 w_1 + (\Delta p)_e + \delta_2 w_2$, where $w_i = \left.\frac{dp}{dx}\right|_i$ is the local pressure drop gradient along with the segment i, δ_i is the length of this segment, and $(\Delta p)_e$ is the pressure drop due to the expansion. This equation corresponds to momentum conservation. Besides, we have to postulate the relation $(w_1, w_2, (\Delta p)_e) = \boldsymbol{H}(q)$.
  4. The combination of the two previous ones. We try to learn Δp from flow q but by means of local pressure gradients as before. Besides, we postulate $(w_1, (\Delta p)_e, w_2,) = \boldsymbol{H}(v_0, v_1, v_2)$ and $(w_1, (\Delta p)_e, , w_2) = \mathsf{H}(v_0, v_1, v_2)$ where we have defined a new set of internal variables, v_i, that must satisfy mass conservation equation, $v_i \Sigma_i = q$.
  5. We try to learn Δp from the flow q as in the previous example. The only difference is that, now, we define $(w_1, (\Delta p)_e, w_2) = \boldsymbol{H}(v_0, v_1, v_2)$ with:

$$H_1(v_0, v_1, v_2) = \lambda\gamma\left(\frac{v_0\Sigma_1}{\kappa_1}\right)^{\alpha}\Phi_1^{\beta}, \tag{14a}$$

$$H_2(v_0, v_1, v_2) = \frac{1}{2}\rho\left[(v_2^2 - v_1^2) + \xi(v_2 - v_1)^2\right], \tag{14b}$$

$$H_3(v_0, v_1, v_2) = \lambda\gamma\left(\frac{v_2\Sigma_2}{\kappa_2}\right)^{\alpha}\Phi_2^{\beta}. \tag{14c}$$

Now, we can establish as unknown parameters $\lambda_1 = \xi$, $\lambda_2 = \frac{\lambda\Phi_1^{\beta}}{\kappa_1^{\alpha}}$ and $\lambda_3 = \frac{\lambda\Phi_2^{\beta}}{\kappa_2^{\alpha}}$. We have therefore $\boldsymbol{H} = \boldsymbol{H}(v_0, v_1, v_2; \boldsymbol{\lambda})$. This is quite common, as it means that we do not know the roughness of each pipe segment (equivalently, the value of $1/\kappa_i^{\alpha}$), but we do know the model and the geometry. The rest of the parameters λ, Φ_i^{β} act as fixed multiplicative constants.

6. The same problem as before, except for the fact that, now, we have several possible models and we want to select the best. For example, using the Darcy–Weisbach model instead of Hazen–Williams' for the distributed losses, the $\boldsymbol{H}$ model writes:

$$H_1(v_0, v_1, v_2) = \frac{1}{2}\rho f_{D_1}\frac{v_1^2}{\Phi_1}, \tag{15a}$$

$$H_2(v_0, v_1, v_2) = \frac{1}{2}\rho\left[(v_2^2 - v_1^2) + \xi(v_2 - v_1)^2\right], \tag{15b}$$

$$H_3(v_0, v_1, v_2) = \frac{1}{2}\rho f_{D_2}\frac{v_2^2}{\Phi_2}. \tag{15c}$$

For that case, the unknown parameters are $\lambda_1 = \xi$, $\lambda_2 = f_{D_1}$ and $\lambda_3 = f_{D_2}$. As f_{D_i} depend on the flow regime, assuming, for example, laminar regime, we have $f_{D_i} = 64\nu/(v_i\,\Phi_i)$.

- **Characterization problem:**

Now the aim is to characterize some of the parameters of the pipe segments for a given set of values (q, p_0, p_1, p_2). That is, the parameters of the constitutive equation vary from one sample to another, and the final goal is to predict those parameters defining the intrinsic behavior of the system, assuming a given state model structure. Let us suppose, for instance, that we want to characterize the roughness of the pipe in terms of the two parameters κ_1 and κ_2. For the sake of simplicity, let us fix a constant area $\Sigma_1 = \Sigma_2 = 1\,\mathrm{m}^2$, equivalent to assume $\xi = 0$ and $\rho = 0$ at Eqs. (14) and (15). Note that this example is very illustrative in the sense that it characterizes a spatially variable property of a given material. This may be extrapolated to obtain the profile of a material parameter $\kappa = \kappa(\boldsymbol{x})$ for heterogeneous materials, when monitoring its behavior under certain actions.

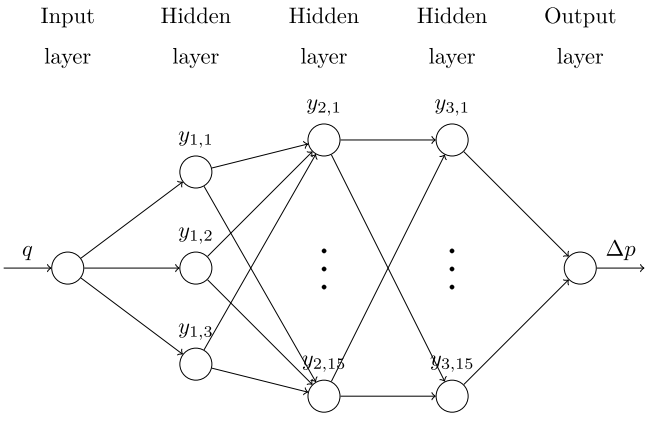
As we are in the heterogeneous case, for discovering the roughness parameters, it is necessary to measure the pressure drop at the two segments, otherwise, the problem would be undetermined. For the present problem, the relation to be learned is $(q, p_0, p_1, p_2) \rightarrow (\kappa_1, \kappa_2)$. For the Hazen–Williams model, the parameters κ_1 and κ_2 are related to the above variables by:

$$\kappa_i = \left(\gamma\lambda\Phi_i^{\beta}\delta_i\right)^{1/\alpha} q\,(p_{i-1} - p_i)^{-1/\alpha}, \tag{16}$$

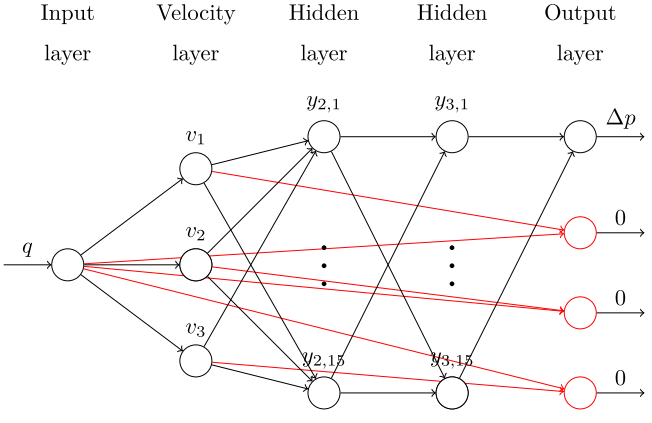
that is $\kappa_i = \lambda_{1,i} q\,(p_{i-1} - p_i)^{\lambda_2}$ where $\lambda_{1,i} = \left(\gamma\lambda\Phi_i^{\beta}\delta_i\right)^{1/\alpha}$ and $\lambda_2 = -\frac{1}{\alpha}$. Note that the parameter dependence is $\kappa_1 = Y(p_0, p_1)$ and $\kappa_2 = Y(p_2, p_1)$. This is not the general case, but could be exploited in the design of the state equation model $\boldsymbol{\kappa} = \mathsf{H}(p_0, p_1, p_2)$, as pointed out in Section 2.3. However, this discussion is important and will be further elaborated in upcoming papers. Here, conventional multilayer perceptrons H will be used to model the state equation $\boldsymbol{H}$.

Depending on the selected approach, we would like:

1. To learn the relationship $(\kappa_1, \kappa_2) = \mathsf{Y}(q, p_0, p_1, p_2)$.
2. To learn the variables κ_1 and κ_2 from the velocities and pressures, $(\kappa_1, \kappa_2) = \mathsf{Y}(v_1, v_2, p_0, p_1, p_2)$, with velocities satisfying the conservation equation, $q = \Sigma_i v_i$.
3. To learn κ_1 and κ_2 from the pressures and flow velocities, $(\kappa_1, \kappa_2) = \mathsf{Y}(v_1, v_2, w_1, w_2)$ where $(w_1, w_2) = \mathsf{H}(p_0, p_1, p_2)$, with w_1, w_2 the head pressure drops (related to viscous forces) and $R(w_1, w_2, p_0, p_1, p_2) = (\delta_1 w_1 - (p_1 - p_0), \delta_2 w_2 - (p_2 - p_1))$ comes from the momentum conservation.



(a) Unconstrained neural network.



(b) Physically-Guided neural network.

Fig. 4. Comparison of unconstrained and constrained neural networks. The constrained neural network (Physically-Guided Neural Network) is illustrated by its physically augmented network, where constraints have been replaced by extra outputs.

3.2. Results of the prediction approach

3.2.1. Direct input–output prediction. Comparison between the classical unconstrained and the constrained neural networks

Let us first consider the problem of predicting directly the nonlinear relationship $q \rightarrow \Delta p$ without any additional constraint, which is a standard NN approach. A neural network is established to solve the *single input–single output* $q \rightarrow \Delta p$ problem proposed. We choose a neural network with only three hidden layers of $n_1 = 3$, $n_2 = 15$ and $n_3 = 15$ neurons, respectively. The network is illustrated in Fig. 4(a)

The neural layers are mathematically defined as follows. Given $\boldsymbol{x} = \boldsymbol{y}_0 = q$, $\boldsymbol{y} = \boldsymbol{y}_4 = \Delta p$ and

$$\begin{aligned} &\boldsymbol{y}_1 = \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1, \qquad \boldsymbol{y}_2 = \boldsymbol{y}_1\boldsymbol{W}_2 + \boldsymbol{b}_2, \\ &\boldsymbol{y}_3 = \mathrm{ReLU}(\boldsymbol{y}_2\boldsymbol{W}_3 + \boldsymbol{b}_3), \\ &\boldsymbol{y}_4 = \boldsymbol{y}_3\boldsymbol{W}_4 + \boldsymbol{b}_4, \end{aligned} \tag{17}$$

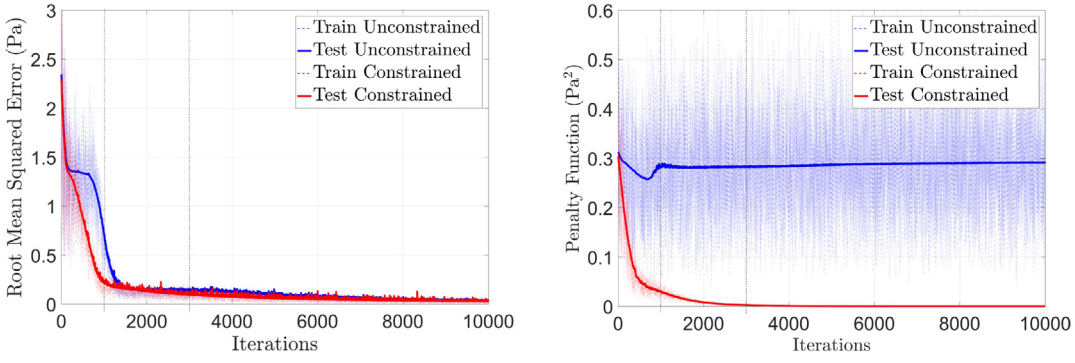
with ReLU a Rectified Linear Unit activation function.

Now we define an analogous neural network in which we impose mass conservation, that is, constant flow in the three reference points of the pipe:

$$v_i \Sigma_i = q, \quad i = 1, 2. \tag{18}$$

Table 1
Physical parameters for the problem with fixed geometry.

Parameter	Σ_1	Σ_1	ρ	ξ	g	κ_1	κ_2	δ_1	δ_2
Value	$1.0\,\mathrm{m}^2$	$2.0\,\mathrm{m}^2$	$1.0\,\mathrm{kg/m}^3$	1.0	$9.81\,\mathrm{m/s}^2$	140	140	10 m	10 m



(a) Root Mean Squared Error (RMSE $= \sqrt{\mathrm{MSE}}$). (b) Penalty function (PEN).

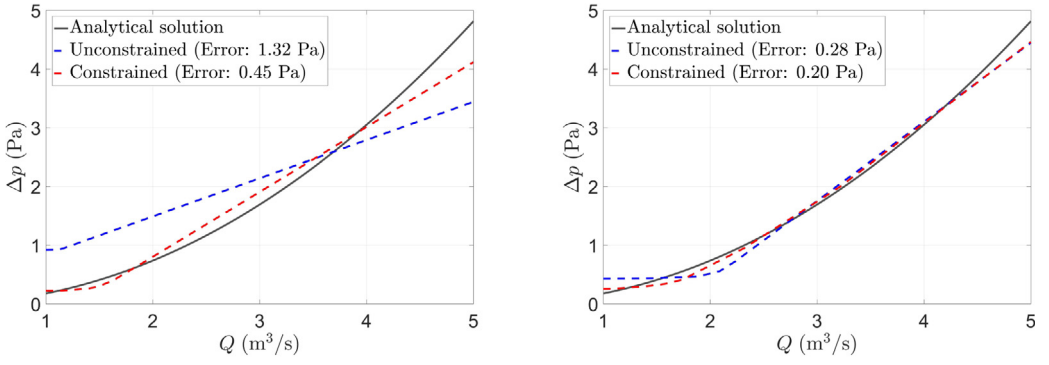
Fig. 5. Root Mean Squared Function and penalty function. Note that, as the topology and parameters of the network between the v and Δp layers are the same, the asymptotic trend of both RMSE functions is the same. Only a modification in the network topology associated with the model, that is, in $\boldsymbol{H}$, would improve the accuracy of the method. The main difference between the networks is the evolution of the value of the penalty function: while for the unconstrained network no penalty term associated to the fundamental physics is added to the loss function, for the constrained network, the penalty term in the loss term (OF) ensures the fulfillment of the constraint, providing physical meaning to the neurons of some internal layers.

The imposition of Eqs. (18) gives the hidden variables $\boldsymbol{y}_1$ a clear physical interpretation: the flow velocities v_i. Eq. (18) is imposed in the neural network via constraints between the values of the corresponding neurons by a penalty approach. Fig. 4(b) illustrates the interpretation of this physically guided network. The network topology is identical to the one of the unconstrained network, but the objective function includes now additional terms, accounting for the constraints associated with the physics.

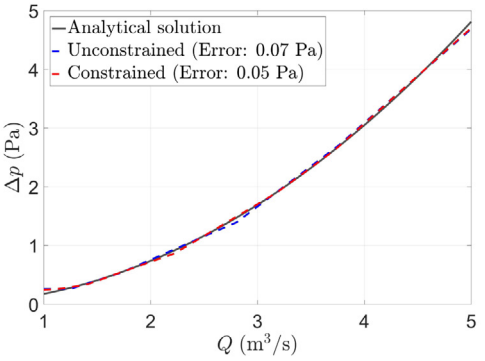
For the training process, the data input was randomly generated with a uniform distribution, using the state model presented in Eq. (12) for $q \in [1.0; 5.0]\,(\mathrm{m}^3/\mathrm{s})$. The physical parameters used for the data generation are shown in Table 1.

As a learning algorithm, a gradient descent optimizer was selected with learning rate parameter $\beta = 0.001$. At each training step, $n = 4$ data points were selected, enough for our purpose. For the constrained network, we chose a penalty parameter $p = 0.01\,\mathrm{Pa}^2\,\mathrm{s}^2/\mathrm{m}^6$. $N_{\mathrm{test}} = 1000$ samples were randomly generated for the testing procedure.

Fig. 5 shows the value of the Root Mean Square Error (RMSE) and the Penalty (PEN) functions along with the training iterations. The effect of including the penalty term (related to the physics of the problem) is clearly illustrated in Fig. 5 that shows that the RMSE has a faster decay in the early learning steps. Indeed, the error converges to the same value when the number of iterations increases. Although there is not a general recipe for the model improvement and each benchmark problem requires its own strategy, as discussed in the broad bibliography on neural networks [49], the fundamental conclusion here is that the constrained network does not necessarily improve the accuracy of the model if they both have the same network topology and metaparameters, but the introduction of physical constraints does speed-up the network convergence. This speed-up is also explained by the evolution of the penalty term value PEN (Fig. 5(b)) for both neural networks: for the unconstrained one, this term is not included in the penalty function and therefore it is not necessarily decreasing. Of course, since the unconstrained network does not force its fulfillment and the error MSE goes also to zero, it is clear that with this network topology, there is not a global minimum solution but many local minima. However, the convergence to one of these minimal solutions is accelerated in the constrained case and for, let us say, $N = 600$ iterations, the behavior of the unconstrained network is suboptimal. Unless the whole underlying parametric model is assumed as known, $H = H(\cdot, \boldsymbol{\lambda})$, what would place us in a case analog to classical parametric fitting via optimization procedures,



(a) $N = 1000$. (b) $N = 3000$.



(c) $N = 10000$.

Fig. 6. Comparison of the network output with the analytical model. Both models present a similar error. The effect of the constraints is not to improve the accuracy, but to speed-up the convergence, besides the physical interpretation of some of the internal layers.

there is no simple way of defining a reduced enough network to guarantee both the required abstraction capability (generalization) and global minimal requirements (specificity), reducing the computational cost.

Fig. 6 shows the accuracy of constrained and unconstrained neural networks after $N = 1000, 3000, 10000$ iterations (marked with a dashed bar in Fig. 5)) when compared to the analytical solution. As explained before, the performance of both networks, if we assume convergence, is similar and only the learning rate, not the accuracy, is improved by the constrained network. The error included in Fig. 6 was computed as:

$$E_{L_2} = \left(\int_0^{10} (Y(q) - Y_m(q))^2 \, dq \right)^{1/2}, \tag{19}$$

where $Y(q)$ is the network predicted pressure drop and $Y_m(q) = \lambda_1 q^2 + \lambda_2 q^{\lambda_3}$ the analytical solution.

The problem may be enriched by taking into account some geometrical aspects. For example, we can consider the two pipe lengths, l_1 and l_2 (until now, they were denoted as δ_1 and δ_2 because they were constant parameters) as extra input variables. This adds a double benefit: (i) it allows us to consider variable geometries and (ii), if some problem parameters are known, the neural network may be simply adapted and simplified to include more physical knowledge. Indeed, for $\xi = 0$, we know that all pressure drop is associated with the distributed head loss along the two stretches so that the hidden layer may be replaced by a layer with two neurons, whose relationship with the output neuron will be $\Delta p = l_1 y_{3,1} + l_2 y_{3,2}$. With these considerations, the third hidden layer acquires also physical meaning (the local pressure drop per unit length at stretches 1 and 2). It is important to note that the constraint in the first hidden layer is now crucial because the input variables have different dimension and the normalization and the ReLU activation function acting between the first and second layer are complemented by the constraint indicating that the lengths l_1 and l_2 do not influence the flow velocity.

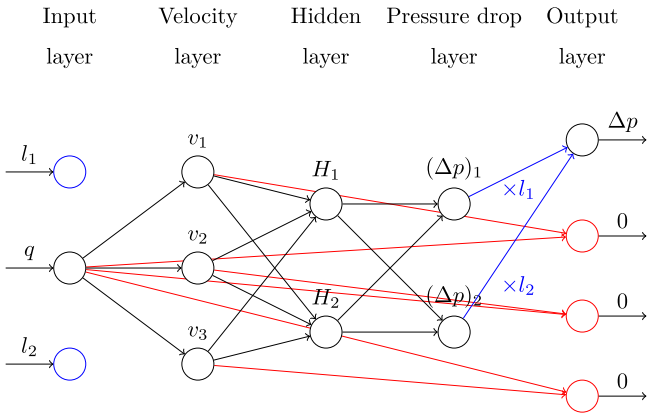


Fig. 7. Physically augmented neural network for the geometry-dependent problem. The red lines illustrate the velocity definition in terms of flow, while the blue lines represent the geometry inclusion by means of the momentum conservation equation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Statistics of the absolute relative error $|\varepsilon_r|$ for the two networks.

Network	Minimum	Q_1	Q_2	Q_3	Maximum	(Mean $\pm$ Std. error)
Unconstrained	2.0×10^{-6}	2.3×10^{-1}	5.0×10^{-1}	8.2×10^{-1}	2.8×10^{2}	$(8.28 \pm 0.03) \times 10^{-1}$
Constrained	4.9×10^{-8}	1.2×10^{-2}	2.1×10^{-2}	4.8×10^{-2}	5.8×10^{-1}	$(4.433 \pm 0.009) \times 10^{-2}$

Neural layers are mathematically defined as follows. Given $\boldsymbol{x} = (q, l_1, l_2)$:

$$\boldsymbol{y}_1 = \mathrm{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1), \qquad \boldsymbol{y}_2 = \mathrm{ReLU}(\boldsymbol{y}_1\boldsymbol{W}_2 + \boldsymbol{b}_2),$$
$$\boldsymbol{y}_3 = \mathrm{ReLU}(\boldsymbol{y}_2\boldsymbol{W}_3 + \boldsymbol{b}_3), \qquad \boldsymbol{y}_4 = y_{3,1}l_1 + y_{3,2}l_2 = \Delta p. \tag{20}$$

The constrained network includes the same constraint as before, relating flow and velocities. The physically guided representation of this new constrained neural network is illustrated in Fig. 7.

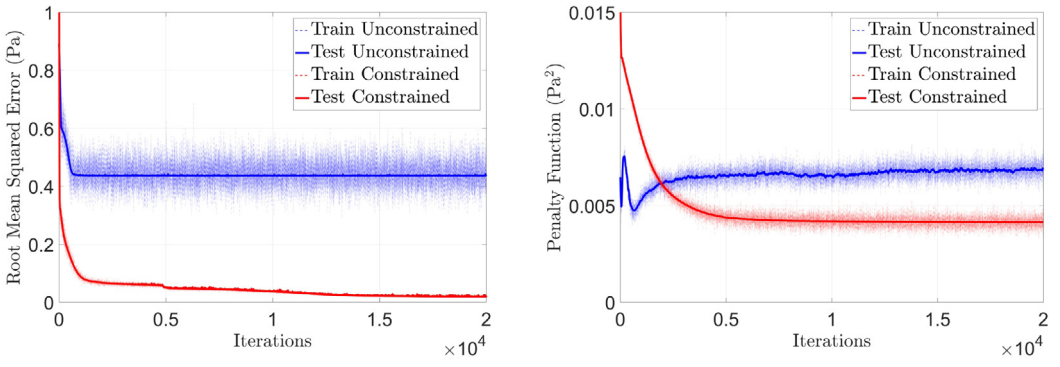
All physical and geometrical parameters are the same as in the preceding example, except that $\xi = 0$, $\Sigma_1 = \Sigma_2 = 1$ m^2, since the effect of the pipe expansion is not the relevant phenomenon here, $\kappa_1 = 140$ and $\kappa_2 = 100$. l_1 and l_2 were uniformly generated between 0 and 10. As before, a gradient descent optimizer was selected with learning rate parameter $\beta = 0.003$. At each training step, $n = 100$ data points were selected. The same value for the penalty parameter ($p = 0.01\,\mathrm{Pa}^2\,\mathrm{s}^2/\mathrm{m}^6$) was selected and $N_{\text{test}} = 10000$ samples were randomly generated for the testing procedure.

As in the previous case, Fig. 8 shows the convergence curves for the RMSE and PEN functions demonstrating good convergence and no overfitting. Fig. 9 shows the accuracy of the constrained and unconstrained neural networks after $N = 20000$ iterations. In that case, in addition to the speed-up of the convergence, the PGNNIV shows a better accuracy, as expected, because the topology of the network was thought in a physical sense, with the last hidden layer having a physical interpretation (the internal variable w_i, $i = 1, 2$).

For comparison purposes, Table 2 shows the statistics of the absolute value of the relative error, $|\varepsilon_r|$, with $\varepsilon_r = \frac{(\Delta p)_{\text{predicted}} - (\Delta p)_{\text{true}}}{(\Delta p)_{\text{true}}}$ obtained for both the unconstrained and constrained networks, when $100 \times 100 \times 100$ values of q, l_1 and l_2 were sampled in $[1; 5] \times [0; 10] \times [0; 10]$, respectively. It is clear that the effect of the constraints is to reduce the relative error together with its variability.

3.2.2. Prediction of the internal variables and identification of the state model

Now, and besides the constraints associated with physical principles, we evaluate the effect of adding model constraints to the network. We shall also discuss the explanatory capacity of the presented method in learning the internal physical variables and, if it is the case, the model parameters. As previously explained, there are two types of equations used in the formulation of the hydraulic head loss in a pipe:



(a) Root Mean Squared Error (RMSE = $\sqrt{\text{MSE}}$). (b) Penalty function (PEN).

Fig. 8. Root Mean Squared Function and penalty function for the network including geometry. The physical constraints give the hidden layers the correct physical interpretation, since the integration constraint, $\Delta p = w_1 l_1 + w_2 l_2$ is correctly formulated. This is achieved thanks to the effect of the penalty term, which gives the PILs layers their correct interpretation. As in Fig. 5, the difference between the constrained and the unconstrained networks is the fulfillment of the constraints after training.

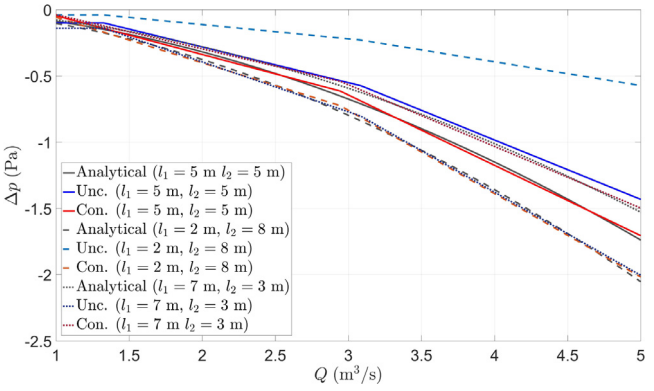


Fig. 9. Exact and predicted solution for different lengths of the segments. The specified network topology and the introduction of the physical constraints are responsible for the network convergence. Results are given for different segment lengths.

- **Fundamental principles**: Mass ($\Sigma_1 v_1 = \Sigma_2 v_2$), linear momentum ($\Delta p = w_1\delta_1 + (\Delta p)_e + w_2\delta_2$) conservation, and energy (hydraulic head) balance ($\Delta h = (\Delta h)_1 + (\Delta h)_e + (\Delta h)_2$) (the subscripts indicate the corresponding segment of the pipe).
- **Constitutive equation**: These equations relate the hydraulic head loss (which is directly related to pressure drop by means of Bernoulli equation) to the fluid velocity along the streamline. For example, the Hazen–Williams or Darcy–Weisbach's for the losses associated with the pipe roughness and Borda–Carnot's for the eddy energy dissipation due to the pipe expansion.

It is clear that there is no general need of including constraints related to the constitutive equation, following the approach of the two previous examples. This is, indeed, contraindicated if there is no knowledge about the underlying behavior of the fluid (physical nature, regime, etc.). However, it might be interesting in at least two circumstances:

1. Model selection: We want to select among many candidate models able to capture, from a macroscopic point of view, the fluid behavior. For instance, in the present example, we shall choose between the Darcy–Weisbach and the Hazen–Williams models for the hydraulic losses. Indirectly, this may give us information about the fluid regime, since the relationship between the Darcy factor f_D and the fluid velocity is fixed ($f_D = \frac{64\nu}{v\Phi}$ for instance for laminar regime).

2. Structure physical discovering: Usually, the model parameters are related to some physical properties that give us insight into the nature, structure, or geometry of the problem. In the present example, for the Hazen–Williams model, κ_1 and κ_2 are related to the roughness of the pipe segments and in the Borda–Carnot model, ξ is related to the smoothness of the expansion.

As the aim is to predict the state model, it is clear that now the output for each data point must be a triplet of values $((\Delta p)_1, (\Delta p)_e, (\Delta p)_2)$ corresponding to the pressure drop at segment 1, expansion and segment 2, respectively. Without this multiple-output consideration, it should be impossible to distinguish between effects in the whole pressure drop. In what follows, three neural networks (with and without constraints) are compared:

1. Model-free approach: Physically-Guided Neural Network where the physics (fundamental laws) are imposed via appropriate constraints in certain layers. This occurs when we add the constraints by means of functions $v_i = E_i(q) = \frac{q}{\Sigma_i}$, $i = 1, 2$.
2. Model-based approach: Physically-Guided Neural Network where both, physical and empirical (constitutive/state equations) laws are imposed.

   (a) Hazen–Williams model: This corresponds to the constraints:

$$(\Delta p)_1 = \lambda \left(\frac{v_1 \Sigma_1}{\kappa_1}\right)^{\alpha} \Phi_1^{\beta} \delta_1, \tag{21a}$$

$$(\Delta p)_2 = \lambda \left(\frac{v_2 \Sigma_2}{\kappa_2}\right)^{\alpha} \Phi_2^{\beta} \delta_2, \tag{21b}$$

$$(\Delta p)_e = \frac{1}{2}\rho q^2 \left[\left(\frac{1}{\Sigma_2^2} - \frac{1}{\Sigma_2^2}\right) + \xi \left(\frac{1}{\Sigma_1} - \frac{1}{\Sigma_2}\right)^2\right]. \tag{21c}$$

   (b) Darcy–Weisbach model: This corresponds to the constraints:

$$(\Delta p)_1 = \frac{\rho}{2} f_{D_1} \frac{v_1^2}{\Phi_1}, \tag{22a}$$

$$(\Delta p)_2 = \frac{\rho}{2} f_{D_2} \frac{v_2^2}{\Phi_2}, \tag{22b}$$

$$(\Delta p)_e = \frac{1}{2}\rho q^2 \left[\left(\frac{1}{\Sigma_2^2} - \frac{1}{\Sigma_2^2}\right) + \xi \left(\frac{1}{\Sigma_1} - \frac{1}{\Sigma_2}\right)^2\right]. \tag{22c}$$

   Note that in that case, for the laminar regime, $f_{D_i} = \frac{64\nu}{v_i \Phi_i}$ and f_{D_i} are constant, while they depend on the pipe roughness in the rough turbulent regime.

In the model-free network, the network topology is prescribed as:

$$\begin{aligned} &\boldsymbol{y}_1 = \boldsymbol{y}_0 \boldsymbol{W}_1 + \boldsymbol{b}_1, \qquad \boldsymbol{y}_2 = \mathrm{ReLU}(\boldsymbol{y}_1 \boldsymbol{W}_2 + \boldsymbol{b}_2), \\ &\boldsymbol{y}_3 = \mathrm{ReLU}(\boldsymbol{y}_2 \boldsymbol{W}_3 + \boldsymbol{b}_3), \qquad \boldsymbol{y}_4 = \boldsymbol{y}_3 \boldsymbol{W}_4 + \boldsymbol{b}_4, \end{aligned} \tag{23}$$

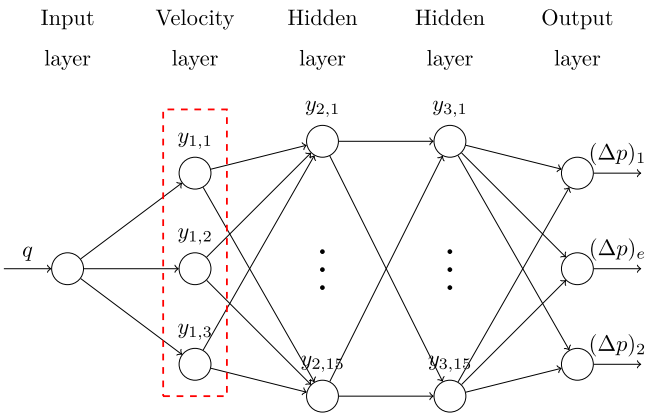
with $\boldsymbol{y}_0 = \boldsymbol{x} = q$ and $\boldsymbol{y}_4 = \boldsymbol{y} = ((\Delta p)_1, (\Delta p)_e, (\Delta p)_2)$. As before, a PIL is prescribed for the variables $\boldsymbol{y}_1$, that will be identified with v_1 and v_2 while the mass conservation is imposed via the constraint $v_i - \frac{q}{\Sigma_i} = 0$. Layers 2 and 3 are composed of $n_1 = n_2 = 15$ neurons.

Similarly, in the model-based network, we propose the following topology:

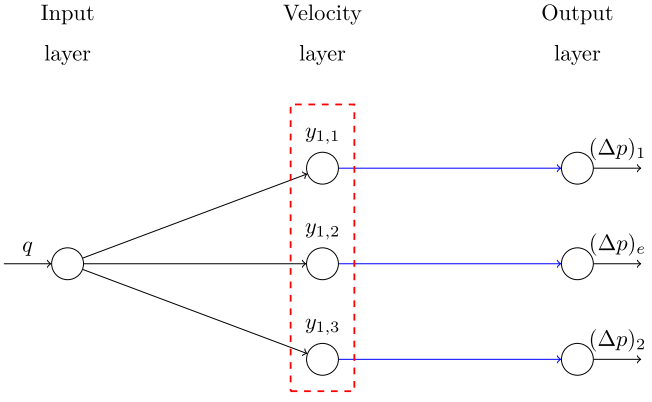
$$\boldsymbol{y}_1 = \boldsymbol{y}_0 \boldsymbol{W}_1 + \boldsymbol{b}_1, \qquad \boldsymbol{y}_2 = \boldsymbol{H}(\boldsymbol{y}_1; \boldsymbol{\lambda}), \tag{24}$$

where $\boldsymbol{H}$ is the model equation, formulated in terms of the model parameters $\boldsymbol{\lambda}$, which are defined as $\lambda_1 = \xi$, $\lambda_2 = \Phi_1^{\beta}/\kappa_1$ and $\lambda_3 = \Phi_2^{\beta}/\kappa_2$ for the Hazen–Williams model and $\lambda_1 = \xi$, $\lambda_2 = \nu/\Phi_1$ and $\lambda_3 = \nu/\Phi_2$ for the Darcy–Weisbach model (in the laminar regime).

Both neural networks are illustrated in Fig. 10.



(a) Model-free neural network.



(b) Model-based neural network.

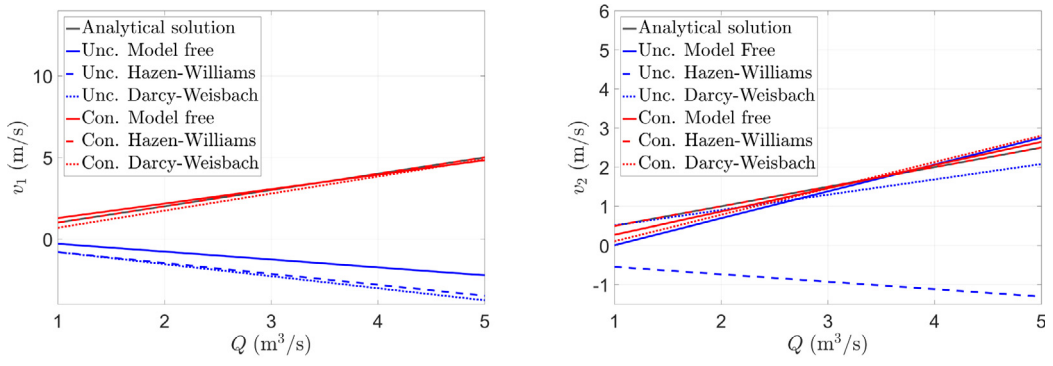
Fig. 10. Model-free and model-based PGNNIV. The universal constraints are illustrated using the red dashed boxes. Model-based constraints are illustrated using blue lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

For the training process, the data input was randomly generated with a uniform distribution using the state model presented in Eq. (12) for $q \in [1.0; 5.0]\,(\mathrm{m}^3/\mathrm{s})$. The physical parameters used for the data generation are shown in Table 1. As a learning algorithm, a gradient descent optimizer was selected with learning rate parameter $\beta = 0.0001$. At each training step, $n = 4$ data points are selected. For the constrained network, we chose a penalty parameter of $p = 0.01\,\mathrm{Pa}^2\ \mathrm{s}^2/\mathrm{m}^6$. $N_{\text{test}} = 1000$ samples were randomly generated for the testing procedure.

To evaluate the performance of all neural networks, we illustrate in Table 3 the statistics of the relative error of the predicted value (when compared to the analytical one) for the different variables involved in the problem: (i) Measurable variables (output variables), that is, the pressure drops $(\Delta p)_1$, $(\Delta p)_e$ and $(\Delta p)_2$; (ii) Non-measurable variables (internal variables), that is, the flow velocity at each segment, v_1 and v_2.

Figs. 11 and 12 illustrate the predictive capacity of the different networks in estimating the internal and measurable variables respectively for different values of q. Once the model-based network has converged, it is possible to extract the model parameters, whose relative error is illustrated in Table 4 and in Fig. 13.

The conclusion drawn is clear and natural. If we want a predictive capability, a model-free neural network is always preferred except if the underlying constitutive model is perfectly known (what is, in general, a strong assumption). A wrong model assumption worsens the network accuracy with respect to a model-free one. Therefore, model-based networks can help in model identification and can shed light on the system's physical and geometrical structure. The specification of an incorrect underlying model affects both correctly specified variables ($(\Delta p)_e$) and those that are not ($(\Delta p)_i$, $i = 1, 2$) as the error is distributed in all the predicted variables.



(a) v_1. (b) v_2.

Fig. 11. Predictive capacity of each neural network in estimating the internal variables. The constrained network is the only one able to predict accurately the internal variables. Model specification improves the accuracy only when the model assumed is the correct one.

Table 3
Statistics of the relative error ε_r for the different networks analyzed. Data is presented as mean value (± Std. error). MF: Model-Free. MB (HW): Hazen–Williams model-based. MB (DW): Darcy–Weisbach model-based. Errors below 1/1000 are not reported.

		Measurable variables			Internal variables	
		$(\Delta p)_1$	$(\Delta p)_e$	$(\Delta p)_2$	v_1	v_2
MF	Unc.	−0.031 (±0.002)	0.030 (±0.002)	−0.070 (±0.002)	1.405 (±0.001)	0.211 (±0.007)
	Con.	−0.049 (±0.003)	0.023 (±0.002)	−0.03 (±0.02)	0.059 (±0.002)	0.096 (±0.003)
MB (HW)	Unc.	−1.000 (±0.000)	0.008 (±0.000)	−1.000 (±0.000)	1.723 (±0.000)	1.666 (±0.004)
	Con.	0.000 (±0.000)	0.000 (±0.000)	0.000 (±0.000)	0.000 (±0.000)	0.000 (±0.000)
MB (DW)	Unc.	−1.000 (±0.000)	0.003 (±0.000)	−0.56 (±0.02)	1.764 (±0.000)	0.120 (±0.001)
	Con..	−0.37 (±0.01)	0.021 (±0.001)	−0.165 (±0.004)	0.097 (±0.002)	0.165 (±0.006)

Table 4
Relative error ε_r of the model parameters for the different model-based networks. MB (HW): Hazen–Williams Model-Based. MB (DW): Darcy–Weisbach Model-Based. Relative error below 10^{-2} (MB (HW) constrained) is marked as 0 in the table because it is of the order 10^{-6}.

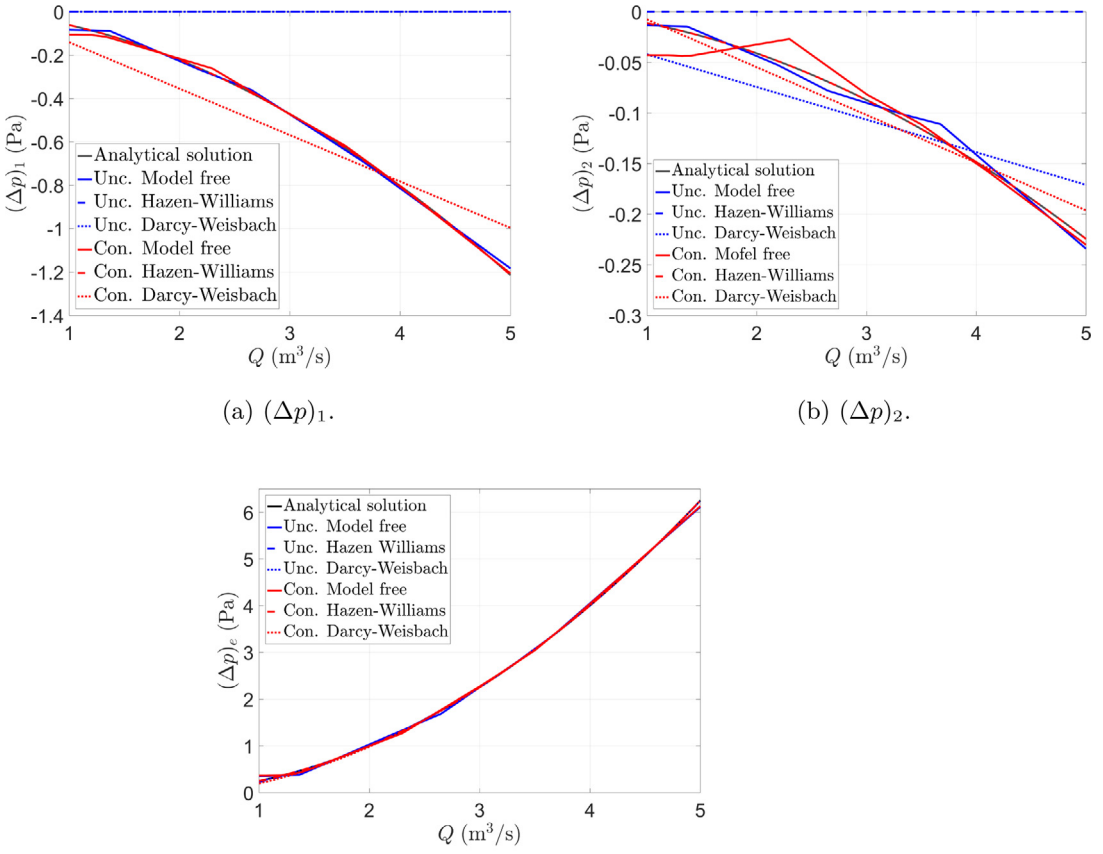
		Parameter		
		λ_1	λ_2	λ_3
MB (HW)	Unc.	1.68	1.00	0.99
	Con.	0.00	0.00	0.00
MB (DW)	Unc.	1.75	1.00	0.58
	Con.	0.42	0.27	0.65

3.3. Results of the characterization approach

Recall that the aim is now to characterize some of the parameters of the pipe segments for a given set of values (q, p_0, p_1, p_2), that is, the relationship to be learned now is $(q, p_0, p_1, p_2) \rightarrow (\kappa_1, \kappa_2)$.

As Eq. (16) is complex and highly nonlinear, it is expected that the number of hidden layers required will be large. Thus, we refer to this part of the network as a *Deep Learning Box* with its own internal number of layers, neurons and connectivity.

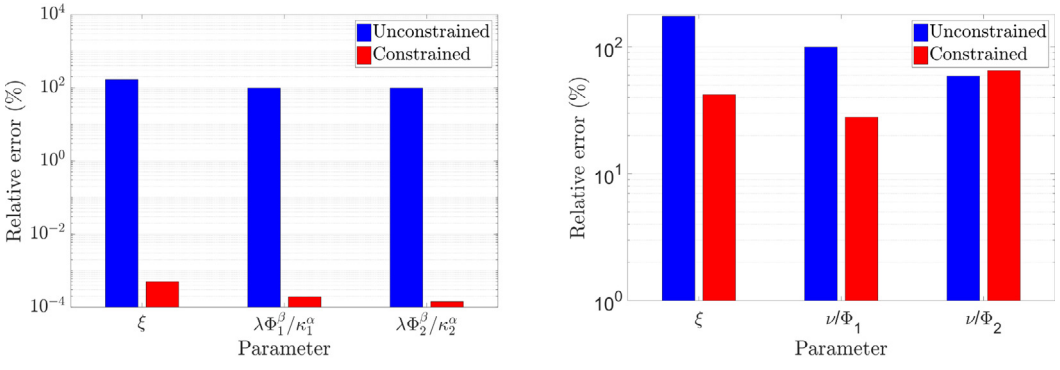
For the problem presented, the PGNNIV topology is shown in Fig. 14. The Deep Learning Box is a multilayer perceptron with 5 dense layers of 20, 40, 80, 40 and 20 neurons respectively, with activation functions of ReLU type. The network performance is compared to the same network in which the physical constraint has not been



(a) $(\Delta p)_1$. (b) $(\Delta p)_2$.

(c) $(\Delta p)_e$.

Fig. 12. Predictive capacity of each neural network in estimating the measurable variables. The unconstrained and constrained networks have a similar capacity in estimating the measurable variables. Model specification improves the accuracy only when the model assumed is the correct one.



(a) Hazen-Williams model-based. (b) Darcy-Weisbach model-based.

Fig. 13. Explanatory capacity of each model-based network. As the data-set was generated using the Hazen–Williams model, only this NN has a perfect explanatory capacity (relative error of the order 10^{-6}) while the Darcy–Weisbach-based network has only partial explanatory capacity.

included. The difference between the constrained and the unconstrained networks is that the penalty parameter is set to zero for the unconstrained network.

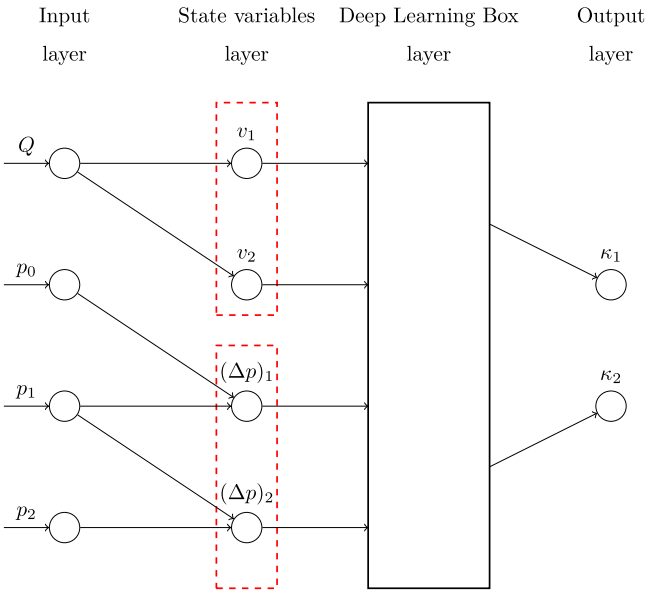


Fig. 14. PGNNIV for the characterization problem. Red dashed rectangles represent physical constraints on neurons. The relationship between flow and velocities is imposed in v_1 and v_2, $v_i = \frac{q}{\Sigma_i}$ and the definition of the incremental pressure drop is imposed in $(\Delta p)_1$ and $(\Delta p)_2$, $(\Delta p)_i = p_i - p_{i-1}$.

The training data-set was created using the analytical model, with κ_1 and κ_2 randomly generated between $\kappa = 80$ and $\kappa = 140$ (that are standard values of the roughness parameter) and a flow q varying from $1\,\mathrm{m}^3/\mathrm{s}$ to $5\,\mathrm{m}^3/\mathrm{s}$. For the training process, we used batches of $n = 300$, a penalty parameter of $p = 0.001$ and a learning rate parameter of $\beta = 1 \times 10^{-5}$ for the gradient descent optimizer. The input x_i and output y_i values are normalized between their maximal and minimal value as:

$$\hat{x}_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \qquad \hat{y}_i = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}}, \tag{25}$$

where $x_{\max}$ and $x_{\min}$ are the maximal and minimal values for the input and $y_{\max}$ and $y_{\min}$ the maximal and minimal values for the output, respectively. $N_{\text{test}} = 100$ test values are used to evaluate the performance.

As for the prediction problem, Fig. 15 shows the performance of both neural networks for the characterization one. As in the previous case, the constraints accelerate the convergence of the network.

The accuracy is shown in Fig. 16 where the predicted values of κ_1 and κ_2 are compared with the theoretical ones for $N_{\text{test}} = 100$ test values. The figure shows a good performance of the neural network although it decays close to the boundaries, what is natural, since the neural network has been trained with data (roughness coefficient) $\kappa \in [80; 140]$.

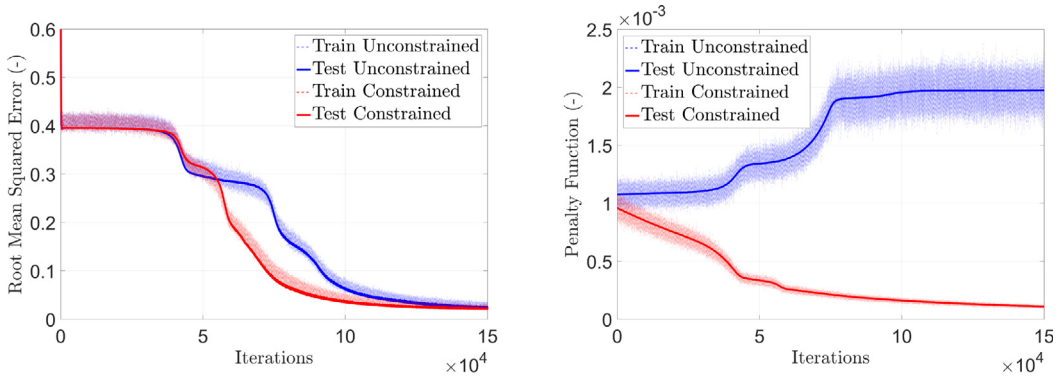
4. Discussion

4.1. Performance improvement

The first important property of the presented methodology, beyond its explanatory capacity, is the improvement of the performance with respect to other Neural-Network-based methods. As PGNNIV has internal constraints between layers (or, equivalently, a higher number of outputs) it is obvious that the search space for the optimal solution will be smaller. This observation leads to four important consequences that are quantified next.

4.1.1. Convergence speed-up

Fig. 17 shows the effect of the constraints in the network convergence for four of the presented problems: (i) fundamental prediction problem (Fig. 17(a)), (ii) problem with the geometry inclusion (Fig. 17(b)), ((iii) model inclusion (Fig. 17(c)) and (iv) characterization problem (Fig. 17(d)). They show the training process for both NNs (constrained and unconstrained) in terms of the Root Mean Squared Error (RMSE).



(a) Root Mean Squared Error (RMSE). (b) Penalty function (PEN).

Fig. 15. Root Mean Squared Function and penalty function. Even if, in all cases, the Deep Learning Box has not enough power to capture the complex nonlinear model perfectly (observe that RMSE does not converge to 0) the effect of the penalty is to speed-up the network convergence.

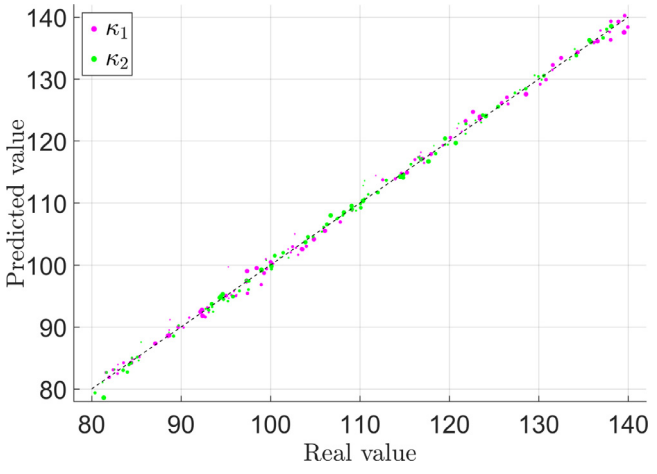


Fig. 16. Results predicted by the PGNNIV. Both predicted values κ_1 and κ_2 are compared to the theoretical ones. Line $y = x$ identifies a perfect estimation. The size of the dot is proportional to the value of the input flow: the model tends to give worse results for smaller flows.

4.1.2. Data need decrease

Another way of seeing the speed-up capability of PGNNIV is to focus on data need. In many engineering problems, especially in those related to material sciences (solid and fluid mechanics, electromagnetism, optics, etc.), there is a lack of experimental data due to technical or economic reasons (*Small Data* framework), so, reducing the amount of training data required in the process is essential. The effect of the constraints will be evaluated in terms of the amount of data required, using the pipe flow prediction problem. For this purpose, the learning curve is evaluated for a varying size of the data-set, $M = 2, 10, 50$. The number of iterations was set as $N = 3000$ and the batch size is fixed to $n = M$ (that is, the whole data-set is evaluated in each training step). The RMSE convergence curves are depicted in Fig. 18. Another way of analyzing these results is illustrated in Fig. 19, where the prediction of $\Delta p = f(q)$ is shown for different data-set sizes and in different convergence steps.

From Fig. 18, it is possible to make some particular conclusions:

- If both networks have the same final accuracy, given a fixed number of iterations, the physically constrained network performance is better (lower error) than the unconstrained one.

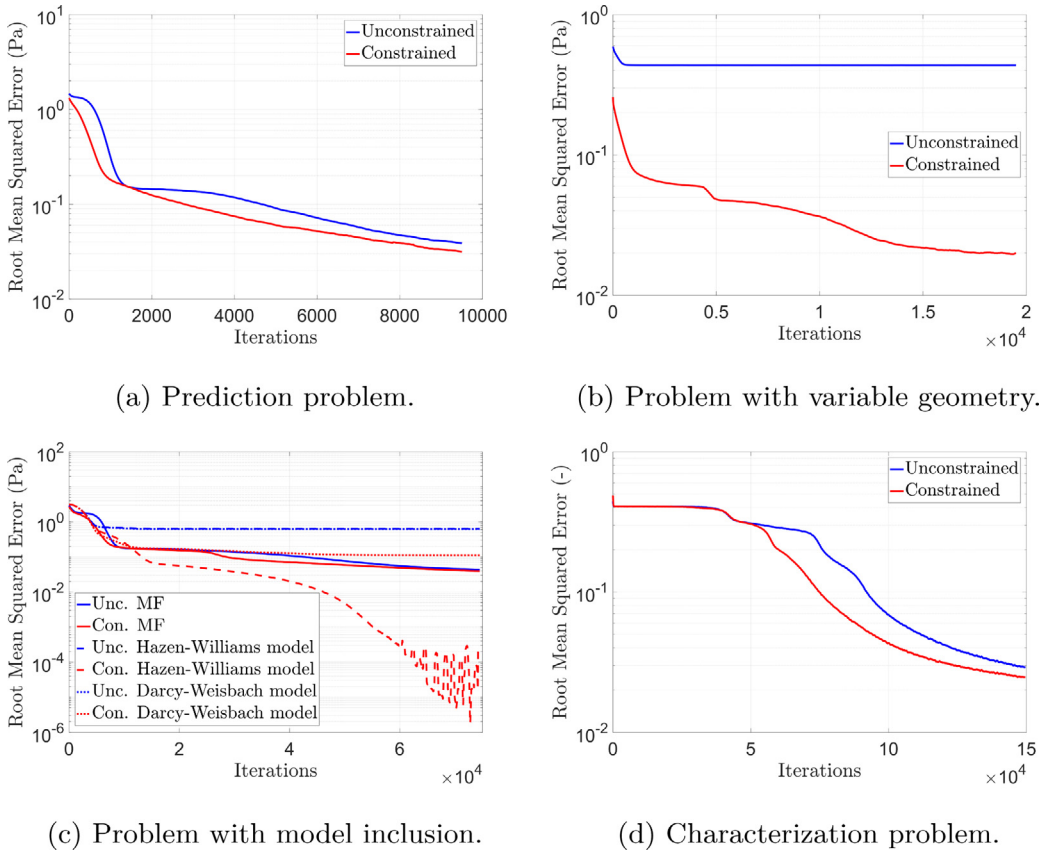


Fig. 17. Convergence comparison of constrained and unconstrained neural networks. Convergence curves are smoothed with a moving average filter (window $W = 500$) for easier comparison. For all cases, the constrained neural network shows a convergence speed-up since the search space is smaller. In general, the accuracy is not necessarily improved but the network convergence is accelerated. Adding constraints (see (c)) always speeds-up the convergence, regardless of wether the accuracy is improved or not.

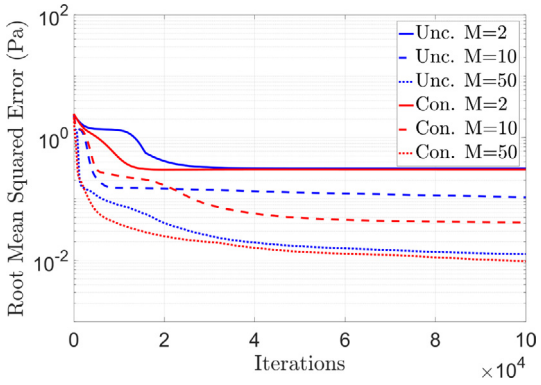
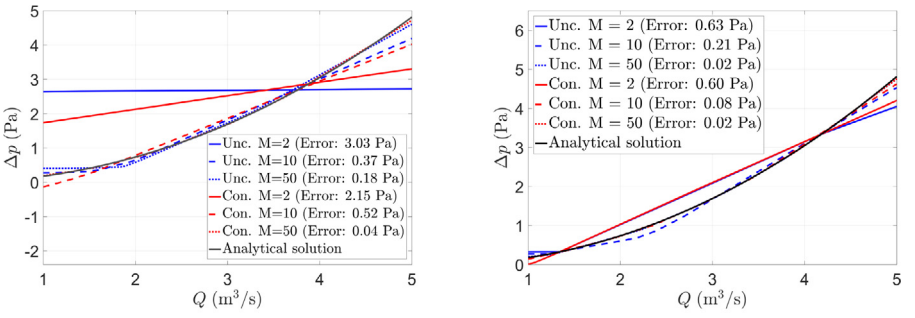


Fig. 18. Learning performance for the two networks and different data-set sizes. Convergence curves are smoothed with a moving average filter (window $W = 500$) for easier comparison. For small data-sets, PGNNIV has an impact not only on the convergence speed-up, but also on the network accuracy.

- The impact of the data-set size is more gradual in the constrained network. Indeed, the curves associated with unconstrained networks are more step-like (specially for small data-sets), when compared to the constrained ones. This is very important in order to detect network convergence stabilization.



(a) $N = 10000$. (b) $N = 100000$.

Fig. 19. Comparison of the network output with the real solution for different data-set sizes and at different convergence states. For a small data-set ($M = 2$), the network only accelerates convergence. When the data-set size is small but large enough ($M = 10$), the constraints have an impact on the network accuracy. For large enough data-sets ($M = 100$), we recover the big data framework, where the constraint accelerates convergence.

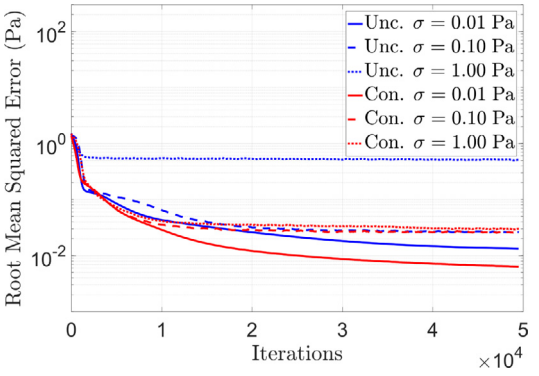


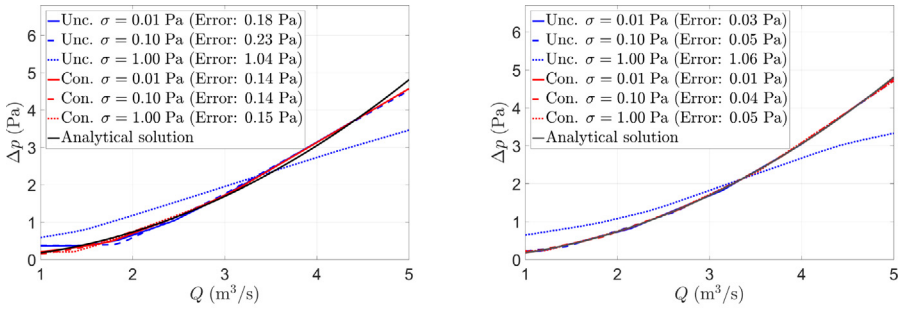
Fig. 20. Learning performance for the two networks and different noise levels. Convergence curves are smoothed with a moving average filter (window $W = 500$) for easier comparison. The noise has a much lower impact in the network convergence, both in speed-up and accuracy.

In brief, constrained neural networks accelerate the learning process in such a way that they are able to discover important new features with less data (small data problems), what is extremely important in a practical engineering context.

4.1.3. Filtering capacity improvement

Here, the noise filtering capacity of the constrained network is explored. In the prediction pipe flow problem, the data-set has been considered noise-free. That is, a data-set $(\overline{q}^i, \overline{\Delta p}^i)$ was generated directly from Eq. (12). Here, we compare the performance of both neural networks when working with noisy data, a more realistic situation in experimental problems and data acquired from sensors. To show the noise impact in the learning process, let us assume a data-sets with an added Gaussian noise, i.e. $x = \bar{x} + Z$ with $Z \sim \mathcal{N}(0, \sigma)$ for $x = q, \Delta p$ and $\sigma = 0.01, 0.10, 1.00\,\text{Pa}$. The RMSE convergence curves are illustrated in Fig. 20. Fig. 21 represents the network accuracy in the $(q, \Delta p)$ plane for different convergence stages.

The effect is even stronger if the constraint acts on the output layer. For instance, let us consider the network with output $((\Delta p)_1, (\Delta p)_e, (\Delta p)_2)$ and let us add the constraint $(\Delta p)_1 + (\Delta p)_e + (\Delta p)_1 = \Delta p$, where Δp is another measured variable (the total pressure drop). In addition to a noise of $\sigma = 0.1\,\text{Pa}$, we consider also the possibility of adding a systematic bias of $-0.2\,\text{Pa}$ to all the measured variables. Fig. 22 shows the convergence curves and Fig. 23 the network accuracy in the $(q, \Delta p)$ plane.



(a) $N = 5000$. (b) $N = 50000$.

Fig. 21. Comparison of the network output with the real solution for different noise levels and at different convergence states. The unconstrained network has a pathological convergence when $\sigma = 1.00\,\text{Pa}$. The constrained network reaches the same solution point for $\sigma = 1.00\,\text{Pa}$ than the unconstrained one with a noise lower in a factor of 10. Besides, at $N = 5000$, all constrained networks provide similar results.

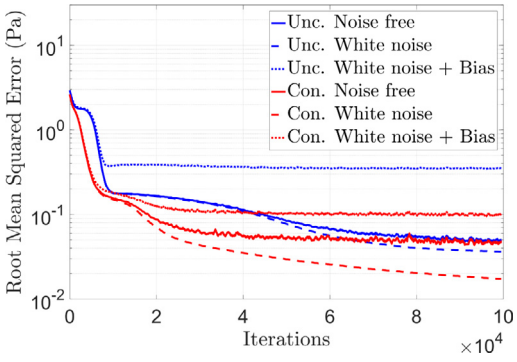


Fig. 22. Effect on the learning performance of white noise and bias. Convergence curves are smoothed with a moving average filter (window $W = 500$) for easier comparison. Bias is partially corrected and noise filtered in constrained networks. Some level of noise may improve the network accuracy, as reported in the literature [50–52].

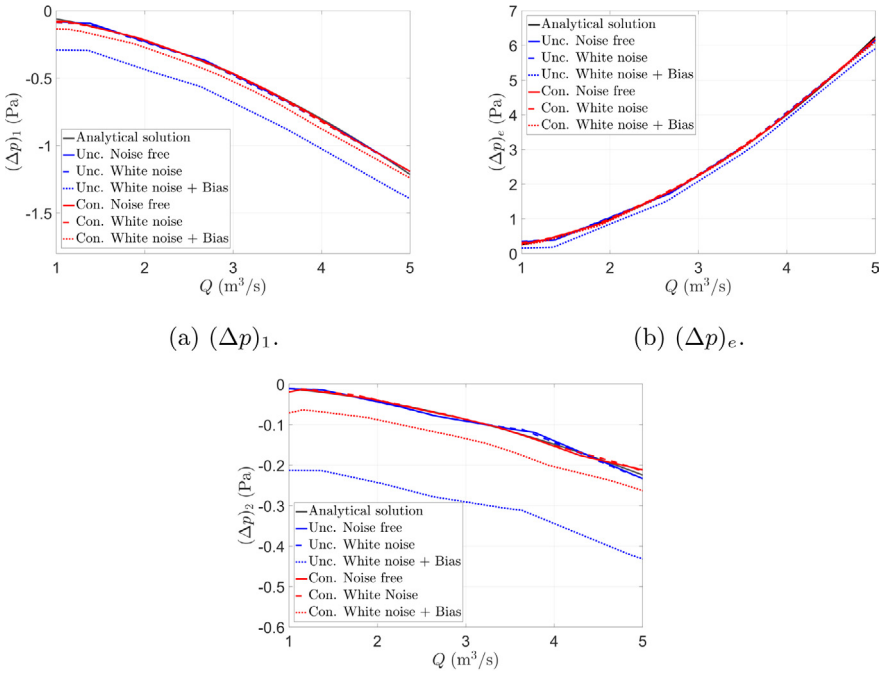
As main results:

- The performance and learning capacity of both networks decrease with noise. This is in agreement with other works [53–55].
- The impact of the noise is lower in the network convergence, as the RMSE curves are closer for the PGNNIV networks.
- The noise has an impact not only in the network convergence rate, but also in the network accuracy, as the curves associated with constrained networks are strictly under the curves associated with the unconstrained ones for $\sigma = 0.01\,\text{Pa}$ and $\sigma = 1.00\,\text{Pa}$. In other words, the physical constraints are able to partially filter the noise.
- Other systematic errors as bias may be corrected partially by the addition of the constraints to the network. That is PGNNIV presents bias partial correction capability.

4.1.4. Extrapolation capability

Let us consider the problem with the constitutive model for head loss estimations. We evaluate the network performance in predicting values of the pressure drops out of the learning data-set, that is, for $q \geq 5$.

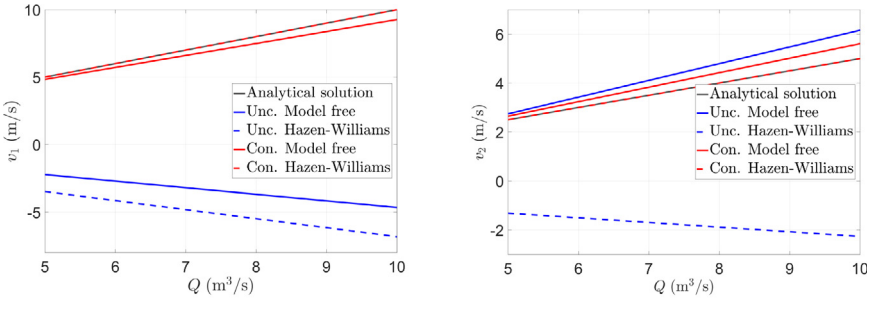
Figs. 24 and 25 illustrate the extrapolation capacity of the different networks in estimating the internal and measurable variables respectively for different values of $q \in [5; 10]\,\text{m}^3/\text{s}$. Fig. 26 shows the relative errors statistics (mean and standard error bar) of the different variables when extrapolating to the new values of q.



(a) $(\Delta p)_1$. (b) $(\Delta p)_e$.

(c) $(\Delta p)_2$.

Fig. 23. Effect of white noise and bias in the network prediction. The effect of adding the constraints in the output layer is a bias reduction in the pressure drop estimation.

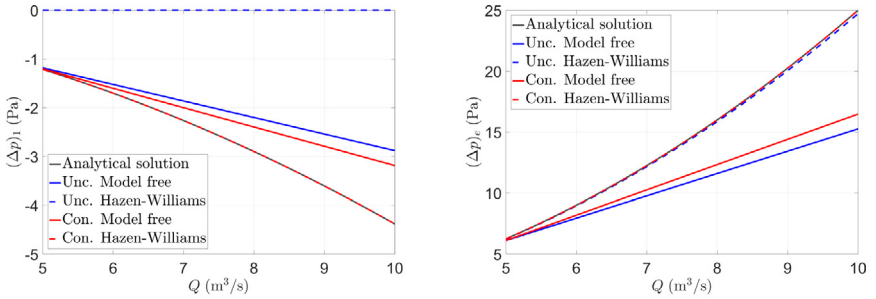


(a) v_1. (b) v_2.

Fig. 24. Extrapolation capacity of each neural network in estimating the internal variables. The constrained network always improves the estimation of the internal variables. For accurate model specifications, the extrapolation capacity is, obviously, total.

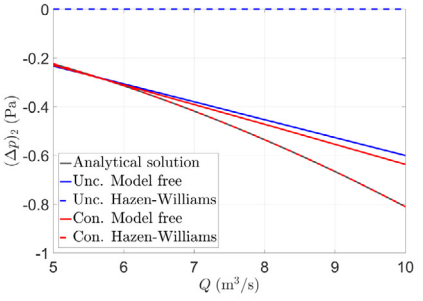
We conclude:

- Even if PGNNIV are designed, among other purposes, for a good estimation of the internal variables, their physically-based nature enables the estimation of the measurable variables out of the learning data-set, which increases their generalization character.
- Model-based PGNNIV networks are, in a certain sense, similar to standard parameter fitting algorithms so if the model is assumed to be known, the converged network has no error in predicting the values following the model assumptions, even for data out of the training range. If we compare PGNNIV, however, to classical fitting procedures, the former has the advantage of all specific hardware and software relative to ANN technology, as explained in Section 2.2.2.



(a) $(\Delta p)_1$.

(b) $(\Delta p)_c$.



(c) $(\Delta p)_2$.

Fig. 25. Extrapolation capacity of each neural network in estimating the measurable variables. The constrained network improves slightly the estimation of the measurable variables. Again, for accurate model specifications, the extrapolation capacity is, obviously, total.

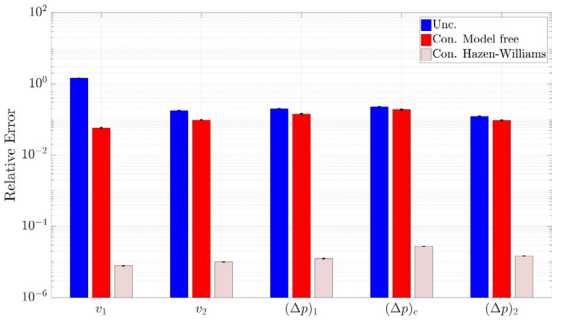
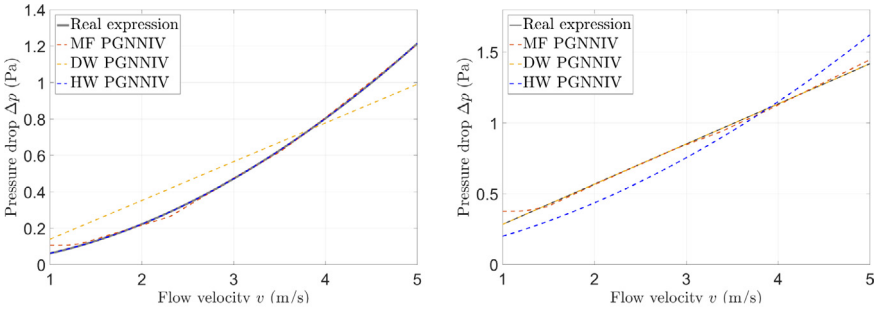


Fig. 26. Evaluation of the extrapolation capacity. PGNNIV always improves the extrapolation capacity, when compared to standard NN. This improvement is significant even for the measurable variables, although the internal variables are of course much better estimated. For model-based networks, the extrapolation capacity is extremely increased.

4.2. Internal state discovery

Despite the characteristics that make PGNNIV faster, less data-demanding and more robust than common NN, these are not the main causes that justify its use in scientific and engineering problems. Indeed, this is due to their physical explanatory capacity. This can be exploited in two ways: (i) accurate prediction of non-measurable internal variables, such as velocities, fluxes or viscous losses and (ii) discovery of the hidden physics in an internal state equation such as the Hazen–Williams'.

The network explanatory capability may be further explored in systems characterization. First, under convergence assumptions, the PGNNIV may be used for predicting the quantitative relation between the different internal variables. But also, the network may provide major features of the structure of the empirical model, for instance, structural dependence and separability.



(a) Hazen-Williams model. (b) Darcy-Weisbach model.

Fig. 27. Model explanatory capacity. MF-PGNNIV has a better explanatory capacity as it is able to provide good results for data-set following different models and to discriminate between them. MB-PGNNIV gives perfect fit when the model is truly the expected one, but gives us worse results if not.

For example, in the prediction problem, we have defined three PGNNIVs: a model-free (MF) PGNNIV, a Hazen–Williams model-based (MB) PGNNIV and a Darcy–Weisbach MB PGNNIV. The presented methodology allows to discover the internal state equation, and to select the best model among several ones. In Fig. 27, we illustrate the relationship $\Delta p = H(v)$, exported from the network after reaching convergence, for the three PGNNIVs and two data-sets, with data derived from Hazen–Williams and Darcy–Weisbach models respectively. It is clear that MF-PGNNIV provides good results for both models, although the two MB-PGNNIV are better suited for the two specific cases. MF-PGNNIV has therefore more explanatory capacity, while MB-PGNNIV have more predictive capacity for the specific cases considered. This is another illustration of the trade-off between explanatory and predictive capacity.

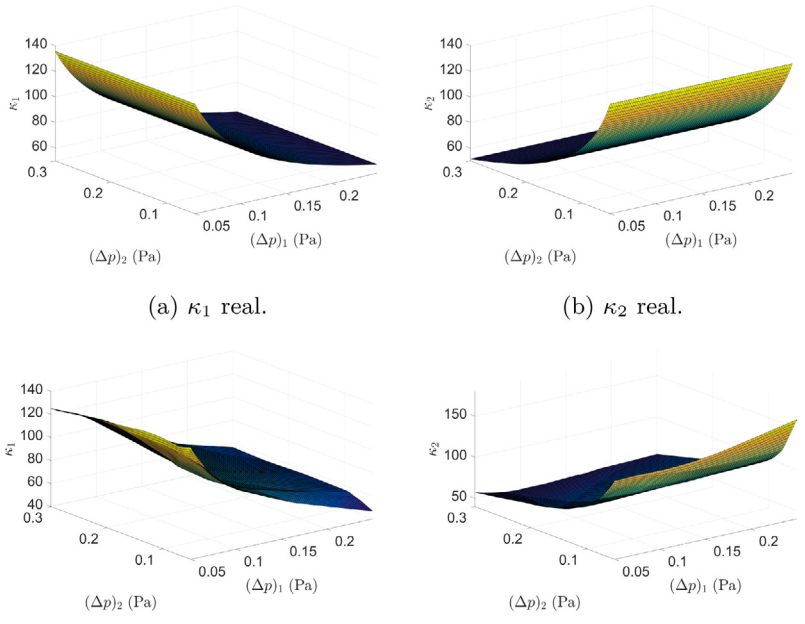
This network explanatory capability may be further explored in characterization also in these two ways: for accurately predicting the $(q, p_1, p_2, p_3) \rightarrow (\kappa_1, \kappa_2)$ relation and also for learning about the model separability. Fig. 28 shows both the real and predicted values for κ_1 and κ_2 for $q = 3$ m^3/s and different values of $(\Delta p)_1$ and $(\Delta p)_2$. As explained in Section 3.3 and may be seen again in Fig. 28, the predicted values are close to the real ones, but a more important fact is that the PGNNIV, thanks to its topology, is able to separate the dependency between variables, that is $\kappa_i = H_i((\Delta p)_i)$ for $i = 1, 2$ instead of the general case $\boldsymbol{\kappa} = \boldsymbol{H}(\boldsymbol{\Delta} p)$. Thus, some features of the model become explainable.

4.3. Link to other methods

In recent years, many Data-Driven methods have been applied to solve problems where some of the physics is known and other has to be discovered. PGNNIV may be compared to the different classes of methods existing in the literature.

F. Chinesta and co-workers use Manifold Learning (ML) to establish the internal state equation $\boldsymbol{\varepsilon} \leftrightarrow \boldsymbol{\sigma}$ [56,57]. In their approach, first the constitutive relationship is computed using Machine Learning techniques in the space $(\boldsymbol{\varepsilon}, \boldsymbol{\sigma})$ (note that $\boldsymbol{\sigma}$ is a non-measurable variable), represented as a low-dimensional manifold. This manifold is then used, instead of the constitutive equation in the problem resolution. Our presented approach is similar in the sense that the model network $\boldsymbol{H}$ may be formulated using the ML framework. Indeed, methods such as kernel Principal Component Analysis (kPCA), Non-Linear Principal Component Analysis (NLPCA), Locally Linear Embedding (LLE) and t-distributed stochastic neighbor embedding (t-SNE) may be formulated in terms of appropriate weights, biases, activation functions and associated hyperparameters and network connectivity.

A very recent idea uses the GENERIC algorithm in time-dependent problems for model identification and evolution prediction [58,59]. This may be seen as a particular PGNNIV, where $\boldsymbol{H}$ is defined using the Poisson and Dissipation operators, $\boldsymbol{L}$ and $\boldsymbol{D}$, together with the discrete version of other differential operators, if necessary. Constraints on many variables may be established, in order to ensure universal physics (the first and second laws of thermodynamics), by means of the degeneracy conditions. Combining this approach with the previous one leads to accurate solutions even while maintaining a reduced computational cost [60].



(a) κ_1 real. (b) κ_2 real.

(c) κ_1 predicted. (d) κ_1 predicted.

Fig. 28. Model explanatory capacity. Values of κ_1 and κ_2 predicted compared to the real ones for different values of $(\Delta p)_1$ and $(\Delta p)_2$ and $q = 3\ \mathrm{m}^3\ \mathrm{s}$.

Two other approaches have been proposed with the same *model-free* philosophy. Other researchers define the constitutive manifold using interpolation instead of regression. The first one is called the *What You Prescribe is What You Get* (WYPiWYG) strategy [61–63] and is based on spline interpolation. The second one is based on nearest-neighbor interpolation, which is totally model-free [64]. Both strategies have demonstrated good performance provided that we have the variables sampled at the space $(\boldsymbol{\varepsilon}, \boldsymbol{\sigma})$. However, these two approaches suffer from extrapolating capacity if the data-set provided has not a broad enough coverage, which is faced in the PGNNIV framework by making flexible the network associated with $\boldsymbol{H}$.

At last and as mentioned, when $\boldsymbol{H}$ is defined via a parametrization of a classical model, $\boldsymbol{H}(\cdot) = \boldsymbol{H}(\cdot, \boldsymbol{\lambda})$, we recover the classical fitting framework (but using neural network tools and algorithms). If, in addition, $\boldsymbol{\lambda}$ is completely specified and the number of weights and biases is less than or equal to the number of parameters $\boldsymbol{\lambda}$, PGNNIV performs merely as a dimensionality reduction.

In a certain sense, PGNNIV framework may be seen as a generalization of all the former approaches. However, only the proposed approach is able to deal with non-measurable variables, albeit performing the data discovering in the state space, where both measurable and non-measurable variables are present. This is possible thanks to the network constraints $\boldsymbol{R}$, from which the state space is built while the unknown internal model $\boldsymbol{H}$ is learned.

5. Conclusions

We have presented a framework in which we use the technology and methods of Artificial Neural Networks (ANN) for solving physically-based problems. This approach allows us both to predict the evolution of a physical system and to explain its structure in the language of Physics.

Of course, it suffers from the typical pros and cons of neural networks. As pros:

- Once the PGNNIV is trained, it allows us to make predictions in an evaluation cost (that is, in real-time). No linear operator inversion, tangent-based operators computation, or iterative procedure is necessary when predicting the state or the evolution of a system.
- The network is trained offline in a very time-consuming process. However, ANN is a mature and hot field in continuous development, and specific hardware and software tools (parallel, cloud and distributed

computation, GPUs and TPUs software, mathematical optimization algorithms, among other software and hardware solutions) are accelerating more and more the training steps.

As main cons, we can mention the following:

- For Deep Neural models, the data used in the training process must be large and varied. This explains why topics such as the Internet of Things (IoT) and the Big Data paradigm are becoming so important in this context.
- Defining neural network successful models needs a complex and time-consuming process of network topology definition and hyperparameter tuning. Although some efforts have been made in the last years for developing appropriate tools [65–68], this remains a hard task that strongly depends on the researcher's knowledge, experience or intuition.
- Even if the ANN converges, it is difficult to expect a prediction to be as accurate as when using fully prescribed mathematical models. PGNNIV are therefore strongly recommended for problems where qualitative explanations or major trends are searched, without entering in fine quantitative details.

In addition to these general characteristics of ANN methods, the presented hybrid formulation has shown extra advantages with respect to other existent methods:

- It allows working only with measurable variables. This is crucial as all the Data-Driven approaches, in one way or another, make hypotheses and assumptions about the relationships between measurable and non-measurable (internal) variables.
- The presented method ranges from model-free (pure prediction) to model-based (explanatory) approaches. In this sense, we talk about model-guided methods.
- As this methodology is physically guided, it allows the explainability of the different phenomena investigated, so it can be framed within the scope of the eXplainable Artificial Intelligence (XAI) [42,69].
- It allows obtaining the whole field of internal variables, without any post-processing of the output variable. This is impossible in any other ANN method without additional assumptions.
- As shown in the presented examples, PGNNIV has both predictive and explanatory capacity. Depending on the aim of the scientist, they can emphasize one capability or the other, depending on their interest, by easily adding/removing constraints, changing the penalty parameters or modifying the network topology.
- Last, but not least, PGNNIV has shown better performance than ANN in aspects such as convergence speed-up, data needs, noise filtering, and extrapolation capacity.

PGNN have just emerged in the last years, but many researchers have come to the conviction that it is the combination of physical knowledge and machine learning tools the appropriate way to adapt the Big Data paradigm to Simulation-Based Engineering and Sciences, overcoming the growing distrust of physical scientists with artificial intelligence.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the financial support from the Spanish Ministry of Science and Innovation (MICINN), the State Research Agency (AEI), and FEDER, UE through the projects PGC2018-097257-B-C31 and PID2019-106099RB-C44/AEI/10.13039/501100011033, the Government of Aragon (DGA) and the Centro de Investigacion Biomedica en Red en Bioingenieria, Biomateriales y Nanomedicina (CIBER-BBN). CIBER-BBN is financed by the Instituto de Salud Carlos III with assistance from the European Regional Development Fund.

References

[1] B. Skuse, The third pillar, Phys. World 32 (3) (2019) 40.
[2] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805, http://dx.doi.org/10.1016/j.comnet.2010.05.010.
[3] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. Hung, Big Data: The Next Frontier for Innovation, Competition, and Productivity, McKinsey Global Institute Reports, 2011.
[4] S. Hill, F. Provost, C. Volinsky, Network-based marketing: Identifying likely adopters via consumer networks, Statist. Sci. 21 (2) (2006) 256–276, http://dx.doi.org/10.1214/088342306000000222.
[5] C.S. Aneshensel, Theory-Based Data Analysis for the Social Sciences, SAGE Publications, Inc., 2013.
[6] W. Raghupathi, V. Raghupathi, Big data analytics in healthcare: promise and potential, Health information science and systems 2 (1) (2014) 1–10.
[7] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, P. Moreno, Automatic language identification using deep neural networks, in: Proceeding ICASSP, 2014, Proceeding of the IEEE International Conference on Acoustic, Speech and Signal Processing, 2014, pp. 5337–5341, http://dx.doi.org/10.1109/ICASSP.2014.6854622.
[8] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceeding NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105, http://dx.doi.org/10.1061/(ASCE)GT.1943-5606.0001284.
[9] D.M. Berry, The computational turn: Thinking about the digital humanities, Cult. Mach. 12 (2011).
[10] S. Leonelli, Introduction: Making sense of data-driven research in the biological and biomedical sciences, 2012.
[11] P. Gould, Letting the data speak for themselves, Ann. Assoc. Am. Geogr. 71 (2) (1981) 166–176.
[12] R. Kitchin, Big data, new epistemologies and paradigm shifts, Big Data Soc. 1 (2014) 1–12, http://dx.doi.org/10.1177/2053951714528481.
[13] R. Shwartz-Ziv, N. Tishby, Opening the black box of deep neural networks via information, 2017, ArXiv abs/1703.00810.
[14] J. Ayensa-Jiménez, M.H. Doweidar, J.A. Sanz-Herrera, M. Doblaré, An unsupervised data completion method for physically-based data-driven models, Comput. Methods Appl. Mech. Engrg. 344 (2019) 120–143.
[15] A. Karpatne, G. Atluri, J. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, A. Kumar, V. Vipin, Theory-guided data science: A new paradigm for scientific discovery, IEEE Trans. Knowl. Data Eng. 29 (10) (2017) 2318–2331, http://dx.doi.org/10.1109/TKDE.2017.2720168.
[16] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[17] X. Lu, D.G. Giovanis, J. Yvonnet, V. Papadopoulos, F. Detrez, J. Bai, A data-driven computational homogenization method based on neural networks for the nonlinear anisotropic electrical response of graphene/polymer nanocomposites, Comput. Mech. 64 (2) (2019) 307–321.
[18] F. Darema, Dynamic data driven applications systems: A new paradigm for application simulations and measurements, in: International Conference on Computational Science, Springer, 2004, pp. 662–669.
[19] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, Science 324 (5923) (2009) 81–85.
[20] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data, 2015, arXiv preprint arXiv:1509.03580.
[21] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[22] P. Ladevèze, The large time increment method for the analysis of structures with non-linear behavior described by internal variables, C. R. Acade. Sci. Ser. II 309 (11) (1989) 1095–1099.
[23] B. Peherstorfer, K. Willcox, Dynamic data-driven reduced-order models, Comput. Methods Appl. Mech. Engrg. 291 (2015) 21–41.
[24] A. Karpatne, W. Watkins, J. Read, V. Kumar, Physics-guided neural networks (pgnn): An application in lake temperature modeling, 2017, arXiv preprint arXiv:1710.11431.
[25] M. Raissi, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.
[26] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.
[27] A. Yazdani, L. Lu, M. Raissi, G.E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLoS Comput. Biol. 16 (11) (2020) e1007575.
[28] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[29] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[30] G. Cybenko, Approximations by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 183–192.
[31] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257.
[32] Z. Lu, H. Pu, F. Wang, Z. Hu, L. Wang, The expressive power of neural networks: A view from the width, in: Advances in Neural Information Processing Systems, 2017, pp. 6231–6239.
[33] B. Hanin, Universal function approximation by deep neural nets with bounded width and relu activations, 2017, arXiv preprint arXiv:1708.02691.
[34] D. Strigl, K. Kofler, S. Podlipnig, Performance and scalability of GPU-based convolutional neural networks, in: 2010 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE, 2010, pp. 317–324.

[35] T.-H. Lee, H. White, C.W. Granger, Testing for neglected nonlinearity in time series models: A comparison of neural network methods and alternative tests, J. Econometrics 56 (3) (1993) 269–290.
[36] W.S. Sarle, Neural Networks and Statistical Models, Citeseer, 1994.
[37] R.-S. Guh, Robustness of the neural network based control chart pattern recognition system to non-normality, Int. J. Qual. Reliab. Manag. 19 (1) (2002) 97–112.
[38] M. McAleer, M.C. Medeiros, D. Slottje, A neural network demand system with heteroskedastic errors, J. Econometrics 147 (2) (2008) 359–371.
[39] J.M. Matías, M. Febrero-Bande, W. González-Manteiga, J.C. Reboredo, Boosting GARCH and neural networks for the prediction of heteroskedastic time series, Math. Comput. Modelling 51 (3–4) (2010) 256–271.
[40] D. Castelvecchi, Can we open the black box of AI?, Nat. News 538 (7623) (2016) 20.
[41] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z.B. Celik, A. Swami, Practical black-box attacks against machine learning, in: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 2017, pp. 506–519.
[42] W. Samek, T. Wiegand, K.-R. Müller, Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models, 2017, arXiv preprint arXiv:1708.08296.
[43] J. Thibault, B.P. Grandjean, A neural network methodology for heat transfer data analysis, Int. J. Heat Mass Transfer 34 (8) (1991) 2063–2070.
[44] R. Hambli, A. Chamekh, H.B.H. Salah, Real-time deformation of structure using finite element and neural networks in virtual reality applications, Finite Elem. Anal. Des. 42 (11) (2006) 985–991.
[45] K. Pathak, S. Panthi, N. Ramakrishnan, Application of neural network in sheet metal bending process, Def. Sci. J. 55 (2) (2005) 125.
[46] J. Weisbach, Lehrbuch der Ingenieur-und Maschinen, Mechanik 1 (1845) 895.
[47] G.S. Williams, A. Hazen, Hydraulic Tables: The Elements of Gagings and the Friction of Water Flowing in Pipes, Aqueducts, Sewers, Etc. as Determined by the Hazen and Williams Formula and the Flow of Water Over Sharp-edged and Irregular Weirs, and the Quantity Discharged, as Determined by Bazin's Formula and Experimental Investigations Upon Large Models, J. Wiley & sons, 1908.
[48] G.K. Batchelor, An Introduction to Fluid Dynamics, Cambridge university press, 2000.
[49] M.A. Nielsen, Neural Networks and Deep Learning, vol. 2018, Determination press San Francisco, CA, USA, 2015.
[50] L. Holmstrom, P. Koistinen, Using additive noise in back-propagation training, IEEE Trans. Neural Netw. 3 (1) (1992) 24–38.
[51] Y. Grandvalet, S. Canu, S. Boucheron, Noise injection: Theoretical prospects, Neural Comput. 9 (5) (1997) 1093–1108.
[52] M. Skurichina, S. Raudys, R.P. Duin, K-nearest neighbors directed noise injection in multilayer perceptron training, IEEE Trans. Neural Netw. 11 (2) (2000) 504–511.
[53] K.-C. Jim, C.L. Giles, B.G. Horne, An analysis of noise in recurrent neural networks: convergence and generalization, IEEE Trans. Neural Netw. 7 (6) (1996) 1424–1438.
[54] E. Kalapanidas, N. Avouris, M. Craciun, D. Neagu, Machine learning algorithms: a study on noise sensitivity, in: Proc. 1st Balcan Conference in Informatics, 2003, pp. 356–365.
[55] D. Rolnick, A. Veit, S. Belongie, N. Shavit, Deep learning is robust to massive label noise, 2017, arXiv preprint arXiv:1705.10694.
[56] E. Lopez, D. Gonzalez, J. Aguado, E. Abisset-Chavanne, E. Cueto, C. Binetruy, F. Chinesta, A manifold learning approach for integrated computational materials engineering, Arch. Comput. Methods Eng. (2016) 1–10.
[57] R. Ibanez, E. Abisset-Chavanne, J.V. Aguado, D. Gonzalez, E. Cueto, F. Chinesta, A manifold learning approach to data-driven computational elasticity and inelasticity, Arch. Comput. Methods Eng. 25 (1) (2018) 47–57.
[58] D. González, F. Chinesta, E. Cueto, Thermodynamically consistent data-driven computational mechanics, Contin. Mech. Thermodyn. 31 (1) (2019) 239–253.
[59] D. González, F. Chinesta, E. Cueto, Learning corrections for hyperelastic models from data, Frontiers in Materials 6 (2019) 14.
[60] B. Moya, D. Gonzalez, I. Alfaro, F. Chinesta, E. Cueto, Learning slosh dynamics by means of data, Comput. Mech. (2019) 1–13.
[61] T. Sussman, K.-J. Bathe, A model of incompressible isotropic hyperelastic material behavior using spline interpolations of tension–compression test data, Commun. Numer. Methods. Eng. 25 (1) (2009) 53–63.
[62] M. Latorre, F.J. Montáns, What-you-prescribe-is-what-you-get orthotropic hyperelasticity, Comput. Mech. 53 (6) (2014) 1279–1298.
[63] V.J. Amores, J.M. Benítez, F.J. Montáns, Average-chain behavior of isotropic incompressible polymers obtained from macroscopic experimental data. A simple structure-based WYPiWYG model in Julia language, Adv. Eng. Softw. 130 (2019) 41–57.
[64] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[65] D. J. Toal, N.W. Bressloff, A.J. Keane, Kriging hyperparameter tuning strategies, AIAA J. 46 (5) (2008) 1240–1252.
[66] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, in: Advances in Neural Information Processing Systems, 2012, pp. 2951–2959.
[67] R. Bardenet, M. Brendel, B. Kégl, M. Sebag, Collaborative hyperparameter tuning, in: International Conference on Machine Learning, 2013, pp. 199–207.
[68] D. Maclaurin, D. Duvenaud, R. Adams, Gradient-based hyperparameter optimization through reversible learning, in: International Conference on Machine Learning, 2015, pp. 2113–2122.
[69] F.K. Došilović, M. Brčić, N. Hlupić, Explainable artificial intelligence: A survey, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2018, pp. 0210–0215.