Physics-Guided Bayesian Neural Networks by ABC-SS: Application to Reinforced Concrete Columns

Juan Fernández[a,*], Juan Chiachío[a], Manuel Chiachío[a], José Barros[b], Matteo Corbetta[c]

[a] *Department of Structural Mechanics and Hydraulic Engineering, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Granada (UGR), Granada 18001, Spain*
[b] *Faculty of Engineering, Catholic University of Santiago de Guayaquil, Guayaquil, Ecuador*
[c] *KBR LLC, NASA Ames Research Center, Moffett Field, 94035, CA, United States*

Abstract

This manuscript proposes a physics-guided Bayesian neural network, which combines *Approximate-Bayesian-Computation* training with physics-based models. This hybrid algorithm uses the laws of physics to mitigate the lack of data, and the flexibility of neural networks to model the complexities inherent in nature. The state-of-the-art approaches often introduce the physics in the loss function, or through some known boundary conditions, and then use backpropagation to adjust the weights. However, this training method involves some rigidity and drawbacks, mostly related to the adoption of a predefined loss/likelihood function and the evaluation of its gradient during training. The use of approximate Bayesian computation as the learning engine results in a greater prediction accuracy and flexibility to quantify the uncertainty, due to the gradient-free nature of the algorithm, the absence of loss/likelihood function and the non-parametric formulation of the weights. Furthermore, the physics-based model is introduced in the forward pass of the neural network, which significantly increases the extrapolation capabilities of the proposed hybrid model. The proposed algorithm has been applied to lateral-load tests in reinforced concrete columns, providing promising results when making predictions about future loading cycles, surpassing the purely data-driven and physics-based methods as well as the state-of-the-art physics-guided neural networks. In light of the performance shown during the experiments, the proposed algorithm has the potential to become a useful tool for fast evaluation of critical buildings after seismic events.

Keywords: Physics-Guided Neural Networks, Hybrid Models, Approximate Bayesian Computation, Uncertainty Quantification, Shear-Capacity Evaluation,

1. Introduction

Physics-based models are mathematical or conceptual representations of some phenomenon, and form the foundations of science and scientific enquiry [1]. From Copernicus' model on the rotation of the planets

*Corresponding author
Email address: juanfdez@ugr.es (Juan Fernández)

around the sun to the most modern quantum mechanics [2], scientists have defined, tested and used models to make predictions, as they approximate reality relatively well and are accessible to human understanding. However, and despite their successful applications, it is complicated for them to include comprehensive details of real natural phenomena without becoming overly complex themselves and difficult to use, with practically unidentifiable parameters [3]. Contrariwise, modern artificial intelligence provides us with algorithms that are capable of learning patterns in complex natural processes without the need to identify and/or understand them, provided that enough data are available [4–6]. And for that same reason, in those situations where data is scarce or imbalanced [7], their performance may be poor and unreliable. Moreover, machine learning algorithms do not perform well when making predictions about events or processes which are outside the training data space (extrapolating) [8]. Unfortunately, there is a wide range of engineering applications where there only exist relatively simple models that partially explain the phenomenon of interest and the availability of data is very limited. Therefore, it seems sensible to seek hybrid models that can benefit from both, physics-based and data-driven approaches.

During the last few years, artificial neural networks (ANN) that include in their loss function some physics-based knowledge about the process that generated the experimental data, such as boundary conditions, have caught the attention of the scientific and engineering community. The way this physics-based knowledge is embedded within the machine learning algorithm is very diverse and depends on the application in hand. Moreover, a deep theoretical understanding of physics and ANN is fundamental for a successful implementation of the algorithm and its hyperparameters. One of the most prominent algorithms in this area of research is the so-called *physics-informed neural networks* (PINN) [9], which encourages the ANN to follow certain laws of physics, described by partial differential equations (PDE), by increasing the cost of solutions that do not satisfy them. This methodology has set the foundations for a wide range of ANN algorithms, such as frictional PINN (fPINN) [10], conservative PINN (cPINN) [11] and extended PINN (XPINN) [12], but also for many applications [13–18]. In this line of research, where ANN are informed by partial differential equations, extensive work has also been undertaken regarding generalization capacity and estimation of the error [19], including XPINN [20]. Another interesting approach to introduce prior domain knowledge in neural networks is by specifying certain constraints, such as logical or algebraic expressions, that should hold over the output space. This method has shown efficiency in computer vision, when mapping from an image to the location of an object it contains [21]. In the area of Prognostics and Health Management (PHM), Nascimiento and Viana proposed a new methodology [22–24] based on a *recurrent cell* where some parts of the physics-based model that are difficult to represent or even measure, such as the stress intensity range in fatigue life prognostics, are compensated by artificial neural networks. Following the same principles, in the field of mechatronic systems (e.g., presses, pumps, hydraulic valves or compressors) the *neural network augmented physics* (NNAP) [25] algorithm proposes neural layers that are inserted in the physics-based model, with the novelty of simultaneously optimizing both the neural network and physical parameters. Physics-guided neural networks (PGNN) are another family of hybrid methods that are providing promising results. Among the different variants of PGNN that can be found in the literature, Jinjiang Wang et al. [26] proposed a cross physics-data fusion (CPDF) scheme to combine the information obtained by a physics-based model and a data driven model for machining tool wear prediction. Ruiyang Zhang et al. [27] presented the Physics-guided Convolutional Neural Network (PhyCNN) for prediction of building's response subjected to earthquakes. Uduak Inyang-Udoh and Sandipan Mishra [28] developed a physics-guided convolutional recurrent neural network (ConvRNN) for droplet-based additive manufacturing and proved that the data required to train this model are much less compared to a full black-box model. Anuj Karpatne et al. [29, 30] was probably the first attempt to introduce some physics-based principles within the neural network architecture, achieving low errors in a lake temperature modeling problem. While most of these algorithms are deterministic in nature, there are some hybrid models that are able to quantify the uncertainty in their predictions, using Bayesian methods [31], arbitrary polynomial chaos (aPC) and *Dropout* [32, 33], or Monte Carlo Dropout [34], among others. However, this quantification of the prediction uncertainty could be defined as *rigid* [35], given that the weights and/or the likelihood function of those neural networks are parametric and defined by a pre-shaped likelihood function, typically Gaussian. In addition, their learning process is based on the evaluation of the gradient of a physics-based loss function via the backpropagation algorithm [36], which may suffer from problems like Dying ReLU [37] or vanishing/exploding gradient [38]. Notwithstanding the foregoing, *Approximate Bayesian Computation by Subset simulation* [39], a method to define posterior distributions of model parameters without having to evaluate likelihoods or gradients, may overcome most of these issues when used as the inference engine to train the weights of ANN [40].

This paper proposes three different methods to combine physics-based models with *Bayesian Neural Networks by ABC-SS* (*BNN by ABC-SS*) , to develop a new physics-guided Bayesian neural network, hereafter called PG-BNN by ABC-SS. The main difference between the proposed algorithms lies in the part of the BNN where the physics-based models are introduced, this is: the metric, the input layer, or the output layer. The gradient-free nature of *BNN by ABC-SS* along with its non-parametric formulation of the probabilistic weights and absence of loss/likelihood function provide great flexibility to capture the uncertainty inherent in the observed data, resulting in a better and less-constrained representation of the reality [40]. While the data-driven part of the proposed algorithm adapts to compensate for the unmodelled conditions in the physics-based model and quantifies the uncertainty, the physics may provide regularization and enables extrapolation. Two different experiments have been carried out: an illustrative problem using synthetic data about projectile motion to evaluate the performance of the proposed algorithm and explain the proposed concept in a straightforward manner; and a real case study on shear strength prediction, using experimental data about lateral-load tests of reinforced concrete columns [41], which constitutes the main application of this paper. The performance of PG-BNN by ABC-SS in both experiments has been compared and benchmarked against their corresponding purely data-driven and physics-based approaches, as well as a state-of-the-art PGNN trained with the backpropagation algorithm using *TensorFlow* [42]. The results clearly show the benefits of combining *BNN by ABC-SS* with physics-based models, such as improved precision and extrapolation capability, and the potential applications of the proposed algorithm. Also, the accurate quantification of the uncertainty shown in the experiments, as a result of using ABC-SS [39] as the learning engine, provides valuable information for any subsequent decision making process.

The rest of the paper is organised as follows. Section 2 provides a brief theoretical background, from the principles and drawbacks of physics-based models and ANN, to the need for hybrid models. Section 3 describes and explains three different proposals on how to introduce the physics-based model into the *BNN by ABC-SS* algorithm. The experimental framework, comprising an illustrative problem and a real case study, is presented in Section 4, including a description of the experimental data, the algorithms used and a discussion on the results obtained. Finally, the conclusions are given in Section 5.

2. Background

This section provides the theoretical background of this manuscript. The principles of ANN are briefly explained in Section 2.1, as well as their main applications and drawbacks. Section 2.2 will introduce *BNN by ABC-SS* along with the importance to accurately quantify the uncertainty. Finally, hybrid neural networks are described in Section 2.3, including the importance of combining physics-based models with data driven methods.

2.1. Neural networks and their drawbacks

ANN have the ability to identify patterns and to extract meaningful information from complex data which other methods cannot process. Some of the main tasks ANN can perform are classification (assigning classes to data points) and prediction (inferring the expected output given an input). This article will focus on the latter, where we train an ANN to make predictions based on observed data. Figure 1 shows a generic representation of an ANN (perceptron), which can be seen as a model f defined by a series of parameters called weights w and bias b. Such model takes some input information $x \in \mathcal{X} \subset \mathbb{R}^n$ and provides an output $\hat{y} = f(x; w, b) \in \mathcal{O}$. The input information flows from the input neurons through the hidden layers to the output neurons. Along that path, the input information is processed and transformed by the weights, bias and the non-linear activation functions in the hidden and output layers. That mapping from inputs to outputs is commonly known as the forward pass, which is shown in Equation 1 for the generic case of Figure 1.

$$\hat{y}_k = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)}) \tag{1}$$

Despite all the powerful attributes of ANN and their applications to a wide variety of problems, they also have some drawbacks that should not be ignored. While we are not interested in listing them all, three of

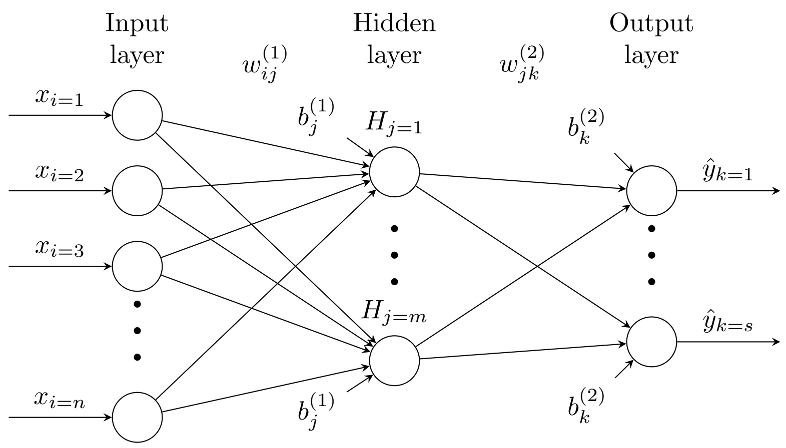


Figure 1: Generic example of a basic Feed-forward Neural Network

them deserve a special mention in the context of this article. Firstly, ANN (in their deterministic form) do not quantify the uncertainty in their predictions, and this can be controversial when there exists a decision to be made based on such predictions. Secondly, ANN are very efficient at making predictions within the domain of the training data, however, in those regions of the data space where no information was available during training (extrapolation), the output of the neural network is unreliable [43] and in most cases it should be discarded. Moreover, it is recommended that the range of the input variables space is recorded during training, so extrapolation can be avoided when making predictions. Paradoxically, our interest often lies in predicting the unknown, hence the importance of improving the extrapolation capabilities of our models. Lastly, ANN often require large amount of training data, which is contrary to most applications in engineering where data is a scarce resource [44]. Nevertheless, these disadvantages are commonly known and the next sections will describe potential solutions, or at least how they can be mitigated in some cases.

2.2. BNN by ABC-SS

As explained in Section 2.1, ANN are widely used to make predictions about a variable of interest, which are subsequently used in a decision making process. Failing to quantifying the uncertainty, or degree of belief, on those predictions can therefore have undesirable consequences. The Bayesian method provides a suitable framework to interpret and quantify the uncertainty in both, the parameters and the outputs of a neural network. Let $\theta = \{w, b\} \in \Theta \subseteq \mathbb{R}^d$ be the parameters of the neural network, namely weights and bias; $\mathcal{M}$ the neural network architecture; and $p(\theta|\mathcal{M})$ the prior information we have about parameters θ, then that prior information can be updated in light of the training data $\mathcal{D}(x, y)$ as follows:

$$p\left(\theta|\mathcal{D},\mathcal{M}\right) = \frac{p\left(\mathcal{D}|\theta,\mathcal{M}\right) p\left(\theta|\mathcal{M}\right)}{p\left(\mathcal{D}|\mathcal{M}\right)} \tag{2}$$

where $p\left(\theta|\mathcal{D},\mathcal{M}\right)$ is the posterior information of the parameters, $p\left(\mathcal{D}|\theta,\mathcal{M}\right)$ is the likelihood function and the term $p\left(\mathcal{D}|\mathcal{M}\right)$ represents how likely data $\mathcal{D}$ is to be reproduced by model class $\mathcal{M}$.

Bayesian neural networks have experienced an increase in popularity in recent times and several variants can be found in the literature, from Variational Inference [45] to Hamiltonian Monte Carlo [46]. However, they often present some disadvantages such as the need to define a likelihood function $p(\mathcal{D}|\theta, \mathcal{M})$ and/or a parametric formulation of the weights and bias $p(\theta|\mathcal{D}, \mathcal{M})$. That results in a rigid representation of the reality, given that the quantified uncertainty is forced to follow a predefined parametric function.

Contrarily, *BNN by ABC-SS* overcomes those problems thanks to the lack of likelihood function and its non-parametric formulation of the weights and bias. In addition, its gradient free nature provides stability and avoids issues like the Dying ReLU [37] or Vanishing/Exploding Gradient [38]. Overall, *BNN by ABC-SS* provides high accuracy rates, similar to PBP and HMC, along with a more precise and realistic quantification of the uncertainty inherent in the observed data. Therefore, *BNN by ABC-SS* is particularly suitable in those cases where quantifying the degree of belief on the predictions is of great importance, as in most engineering problems. The interested reader is referred to [40] for further information about ABC-SS, and to [39] for *BNN by ABC-SS*.

2.3. The raison d'Etre of hybrid neural networks

Physics-based models are the basis of today's technology, and have made possible uncountable achievements. However, the rise of big data and the rapid development of computational capacities have resulted in the popularization of data-driven approaches within the scientific community, such as ANN. In some applications, these data-driven approaches have outperformed physics-based models, especially in those cases where large amounts of data are available and/or the underlying governing laws are unknown. It could even be argued that physics-based models are just an approximation of reality and subject to lack of knowledge or poor understanding of complex mechanisms that may be involved in the phenomenon to be modelled, while observed data is a measurement of something taken directly from reality itself. Those might be some of the many reasons behind this increase in popularity of data-driven methods. As mentioned in Section 2.1, ANN have been very successful indeed in performing many tasks, but they also suffer from significant drawbacks such as their difficulties to generalize and incapacity to extrapolate. Interestingly, those are some of the best known qualities of physics-based models, their ability to make consistent predictions regardless the availability of data to learn from.

Combining physics-based models with ANN into hybrid models is a promising line of research [47, 48]. Specially in some engineering applications, where the known physical laws struggle to accurately model the reality, either because of the complexity of the process to be modelled or simply lack of knowledge. In addition, data is not abundant when dealing with engineering problems like failure prediction, so purely data-driven approaches are not suitable on their own. And in any case, it seems sensible to make use of any knowledge we have at hand, no matter if it comes from observed data or validated physical laws. Some of the engineering disciplines that have shown a promising advance in the use of hybrid models are the energy

[49] and prognostics [50] fields. Figure 2 shows a simplified conceptualization of hybrid models.

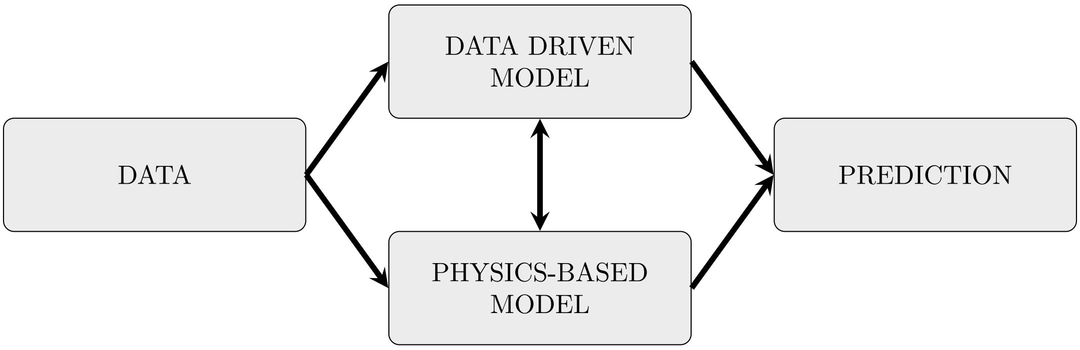


Figure 2: Conceptual chart of hybrid models

3. PG-BNN by ABC-SS

In this section, three different methods are proposed to combine physics-based models with *BNN by ABC-SS* to obtain hybrid Bayesian neural networks, the so-called PG-BNN by ABC-SS. Details of the implementation, changes to the original *BNN by ABC-SS* algorithm and a description of the expected behaviour of the hybrid models are provided. The proposed neural network architectures are also shown graphically so they can be compared against the standard ANN shown in Figure 1 and Equation 1. It should be noted that, depending on the nature of the problem to be solved, the physics-based models described below may be substituted by any model $M(x) = y$, where M represents the model class, x is the input information and y the output of the model.

3.1. Physics learnt through the metric function

ANN base their training and updating of parameters on a loss function, while BNN often use a predefined likelihood function chosen by the user. *BNN by ABC-SS* lacks both of them, and instead approximates the likelihood function to $P\left(\hat{y} \in \mathcal{B}_\epsilon(y)|\theta\right)$, which can be interpreted as the probability of $\hat{y}$ to fall within a region $\mathcal{B}_\epsilon(y) = \{\hat{y} \in \mathcal{O} : \rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon\}$, where a metric function $\rho(\cdot)$ assesses the distance between prediction $\hat{y}$ and data $y \in \mathcal{D}(x, y)$, based on a summary statistics $\eta(\cdot)$ chosen by the user. Thus, a set of parameters $\theta = \{w, b\}$ will be accepted only if $\rho(\eta(\hat{y}), \eta(y)) \leqslant \epsilon$, in other words, only if prediction $\hat{y}$ is close enough to the data y. As mentioned in Section 2.2, this lack of likelihood function provides an enhanced flexibility which results in a fairer representation of the uncertainty [40].

In this work, a new metric function ρ_p based on the laws of physics, is proposed in addition to the current data-driven metric. This will ensure that during training a set of parameters θ is accepted only if, given an input $x \in \mathcal{D}(x, y)$, the prediction $\hat{y}$ is also close enough to the prediction y_p made by the physics-based model. Moreover, two hyperparameters α and β are included in the final metric function, so the user can control how much weight is given to the physics-based model and to the data-driven approach. Algorithm 1 shows the necessary adaptation of *BNN by ABC-SS*, and Figure 3 a schematic representation of the concept.

Algorithm 1 *Physics Learnt Through the Metric Function*

Every time $\rho()$ needs to be calculated in Algorithm 1 of [40], replace it as follows:
New terms:
$\rho_d()$ {data-driven metric}
$\rho_p()$ {physics-based metric}
$\rho_f()$ {overall or final metric}
y_d {data y from training data $\mathcal{D}(x, y)$}
y_p {prediction from physics-based model}
α {weight given to the data-driven approach}
β {weight given to the physics-based approach}
Begin:
$\rho_d \leftarrow \rho(\eta(\hat{y}), \eta(y_d))$
$\rho_p \leftarrow \rho(\eta(\hat{y}), \eta(y_p))$
$\rho_f \leftarrow \alpha\rho_d + \beta\rho_p$

The proposed physics-based metric is expected to provide a regularization effect, which will be added to the natural regularization of BNN thanks to the prior information. This may translate into better generalization, helping to avoid overfitting. Extrapolation might also improve slightly in some cases as the BNN is encouraged to comply with the actual physics, however, since the physics-based model is not implicitly included in the forward pass, accurate extrapolation is not anticipated nor should it be considered as a goal. The uncertainty quantified by the proposed algorithm might also be reduced, given that the BNN now has more information and will discard those data points that depart significantly from the laws of physics.

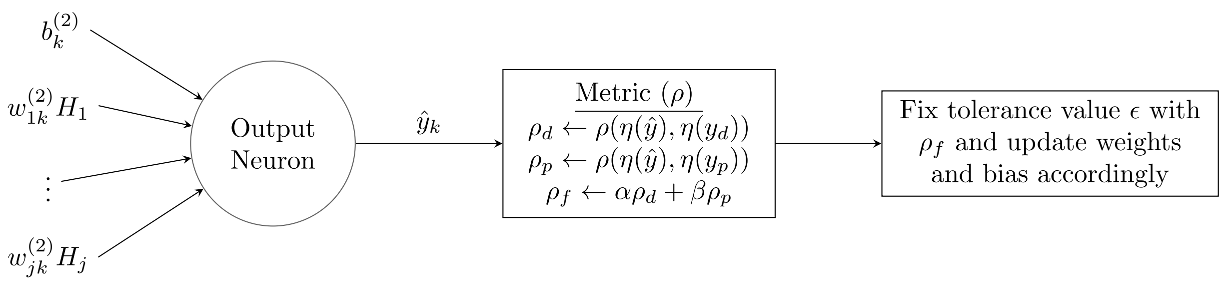


Figure 3: Schematic representation of proposed PG-BNN by ABC-SS

3.2. Physics learnt through input neurons

Principal Components Analysis (PCA) is an interesting method to reduce the dimensionality of big data sets where there exists a large number of variables [51]. Simplistically, PCA aims to find any existing relationship between variables, and then uses linear combinations representing such relationships as new variables. Although reducing the dimensionality of data sets is not the objective of this manuscript, the conceptual idea of PCA about identifying correlations between variables and use them as new inputs could serve as an analogy and inspiration for introducing the laws of physics in neural networks. After all, in engineering applications the input and output variables of the neural network are related by the laws of physics. Figure 4 shows how physics could be embedded into the architecture of the neural network via the input layer. The term *Physics* refers to the output of the physics-based model, and the subscript $p = 1, .., s$ in $Physics_p$ indicates the number of outputs in the physics-based model, in case there are multiple outputs.

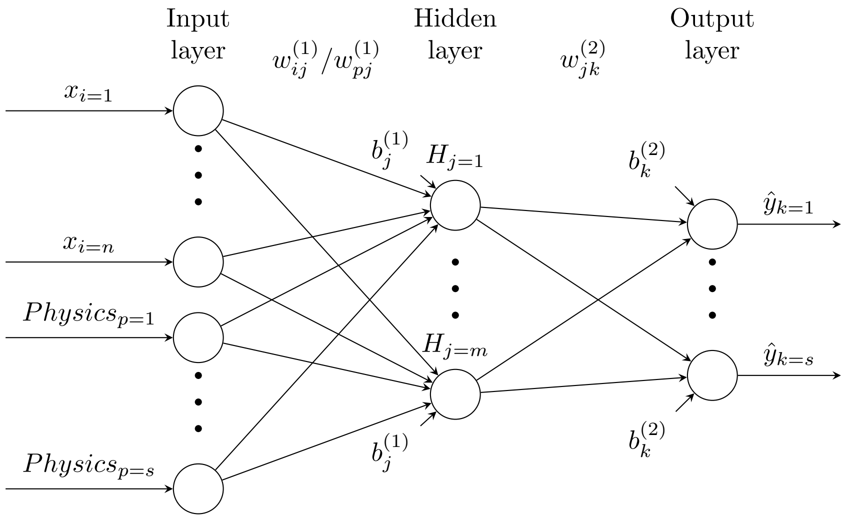


Figure 4: Architecture of PG-BNN by ABC-SS via the input layer

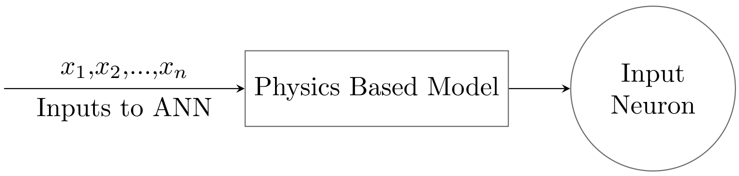


Figure 5: Physics-based model fed into the input neuron

When the output of the physics-based model is used as a new input to the neural network, as in Figure 5, the latter is informed about a relationship between the input and output variables. That information could be comprehensive or incomplete, but it will contribute to the learning process in all cases. Moreover, during training the weights and bias of the neural network will learn how to manipulate and change the physics, based on the inputs, so the predictions of the neural network match the observed data.

The forward pass now includes the laws of physics, as shown in Equation 3. Therefore, this knowledge is also applied to predictions made outside the domain of the training data, thus improving the extrapolation capacities of the neural network. The uncertainty is also expected to reduce, given that the neural network is now better informed. Apart from the forward pass, the algorithm of *BNN by ABC-SS* remains unchanged.

$$\hat{y}_k = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + \sum_{p=1}^{s} w_{pj}^{(1)} Physics_p + b_j^{(1)}) + b_k^{(2)}) \tag{3}$$

3.3. Physics learnt through output neurons

The laws of physics should be able to explain most processes and actions that occur in the real world, however, we see that this is not the case nowadays. Quoting the British statistician George E. P. Box, "All models are wrong, but some are useful". There is much truth in that sentence, but it does not mean that the real world is not governed by physics, but that we are not able to precisely model all the complexities of the real world. In fact, the more complex models are, the more hyperparameters they include and more prone they are to overfitting. Conversely, simple models tend to generalize better and are less sensible to the tuning of the hyperparameters, probably at the expense of making less accurate predictions. This is where ANN may help, given their capacity to learn non-linear patterns from observed data. Therefore, it seems sensible to use ANN to identify and learn those complexities in the real world that simple physics-based models cannot.

As shown in Figure 6, this idea can be materialized by adding the outputs of the physics-based model to the output layer of the neural network, just like another bias parameter (Figure 7). With this new architecture the weights and bias of the neural network are forced to adjust to compensate the information coming from the laws of physics, learning those complexities and patterns that are missing in the physics-based model, such as environmental factors. Furthermore, those complexities do not need to be identified or defined in advance, given that *BNN by ABC-SS* provides great flexibility to adapt to different patterns and capture the uncertainty present in the observed data as a whole, no matter their nature [40, Section 5]. The physics-based model is therefore included in the forward pass as per Equation 4, which will improve the extrapolation capacities of the neural network significantly. Besides, the patterns learnt during training to compensate the physics will be propagated to those unexplored regions of the output variable space. The uncertainty is expected to reduce greatly within the domain of the training data, but also outside of it. Apart from the forward pass, the algorithm of *BNN by ABC-SS* remains unchanged.

$$\hat{y}_k = g(\sum_{j=1}^{m} w_{jk}^{(2)} h(\sum_{i=1}^{n} w_{ij}^{(1)} x_i + b_j^{(1)}) + b_k^{(2)} + Physics_k) \tag{4}$$

4. Experimental framework

Two different experiments have been carried out, an illustrative example which aims to clarify the concepts explained in previous sections using a low-complexity problem, where the results and their transcendence can be easily interpreted; and an real case study, which is the main engineering application of

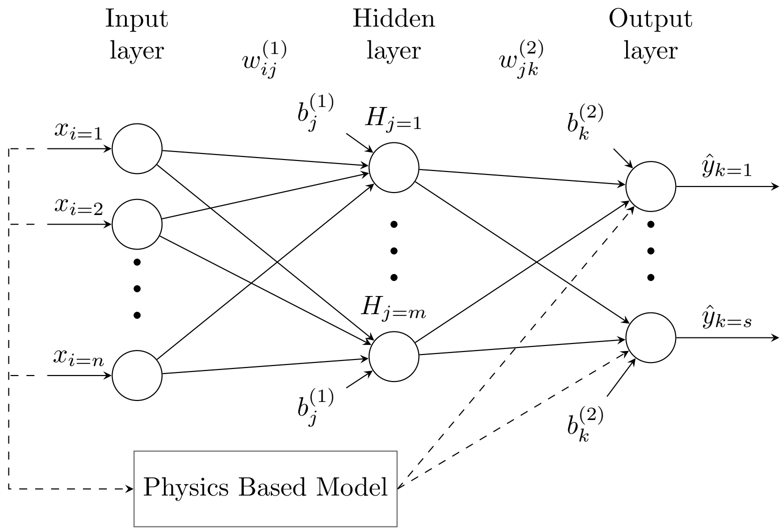


Figure 6: Architecture of PG-BNN by ABC-SS via the output layer

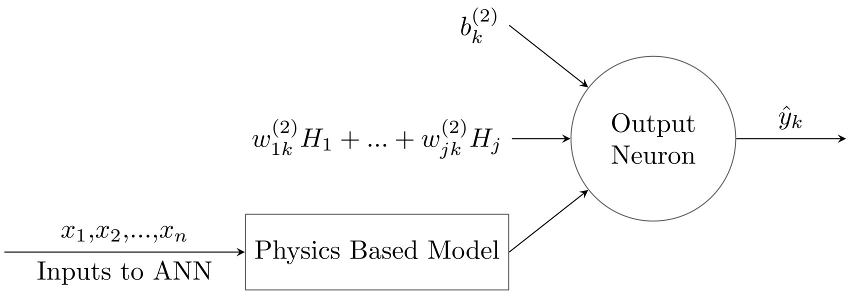


Figure 7: Physics-based model fed into the output neuron like a bias parameter

this manuscript. In this section, the experiments are described including how the data sets are prepared, what algorithms are used, and finally, the results are presented and discussed.

4.1. Formulation of experiments and data preparation

The context of both the illustrative example and the engineering application is given in this subsection, along with information on the source of the data used and how they have been processed for reproducibility.

4.1.1. Illustrative problem: projectile motion

A projectile motion problem, using synthetic data generated in Python, has been chosen to illustrate the concepts presented in Section 3. In this case, a two-dimensional problem is considered where there is no lateral movement, and therefore, the motion along the perpendicular axis 'x' (horizontal) and 'y' (vertical) can be studied independently. The variable of interest (output) is the distance travelled by the projectile d_t, which depends on the initial velocity v_0 of the projectile, the initial angle λ_0 relative to the horizontal assuming a level ground, and some unknown environmental conditions. The non-linear relationship between the independent variables v_0 and λ_0 is given by the following physics-based model:

$$\mathcal{R} = \frac{v_0^2 \sin 2\lambda_0}{g} \tag{5}$$

where g represents the vertical acceleration due to gravity, which is approximated to 9.81 m/s^2.

Finally, some unknown or environmental conditions need to be modelled so the synthetic observed data differs from the pure physics. For this example, some headwind has been added so that the distance travelled by the projectiles is reduced depending on the initial angle λ_0. It has been assumed that, when such angle is less than 45° the distance d_t is reduced by $\mathcal{N}(0.02\mathcal{R}, 1)$, and $\mathcal{N}(0.04\mathcal{R}, 1)$ otherwise. This is to simplistically simulate the surface friction near the ground which forces the wind to slow. That results in projectiles with high initial angle λ_0 and long range $\mathcal{R}$ to be more affected by the wind (∼4% of $\mathcal{R}$) than those with flatter angles and short ranges (∼2% of $\mathcal{R}$). It is worth mentioning that the observed data has been created with the only purpose of illustrating the concept of PG-BNN by ABC-SS and its potential, hence the headwind is just a non-complex pattern that is added to account for some unknown conditions.

Three data sets have been created, one training data set and two test data sets, thus the interpolation (Test Set 1) and extrapolation (Test Set 2) capabilities of the different algorithms can be evaluated. Therefore, the synthetic data is generated as follows:

$$\text{Training Data Set} \begin{cases} \text{300 data points} \\ \text{Inputs: } \lambda_0 \in [30, 60] \text{ and } v_0 \in [30, 60] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R}, 1) \text{ when } \lambda_0 \leq 45° \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R}, 1) \text{ when } \lambda_0 > 45° \end{cases} \end{cases}$$

$$\text{Test Data Set 1} \begin{cases} \text{150 data points} \\ \text{Inputs: } \lambda_0 \in [30, 60] \text{ and } v_0 \in [30, 60] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R}, 1) \text{ when } \lambda_0 \leq 45° \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R}, 1) \text{ when } \lambda_0 > 45° \end{cases} \end{cases}$$

$$\text{Test Data Set 2} \begin{cases} \text{150 data points} \\ \text{Inputs: } \lambda_0 \in [10, 30] \cup [60, 80] \text{ and } v_0 \in [10, 30] \cup [60, 80] \\ \text{Outputs: } d_t \begin{cases} \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.02\mathcal{R}, 1) \text{ when } \lambda_0 \leq 45° \\ \frac{v_0^2 \sin 2\lambda_0}{g} - \mathcal{N}(0.04\mathcal{R}, 1) \text{ when } \lambda_0 > 45° \end{cases} \end{cases}$$

The input vectors are organised in two-dimensional arrays including the initial angle and velocity [λ_0,v_0], while the output vectors are one-dimensional arrays containing the distance travelled by the projectile $[d_t]$. The physics-based information is arranged in one-dimensional arrays $[\mathcal{R}]$.

4.1.2. Engineering case study: application to lateral-load tests in reinforced concrete columns

The engineering application of the proposed PG-BNN by ABC-SS consists of a cantilever reinforced concrete beam-column, subjected to constant axial load and variable cyclic lateral deformation. The lateral force F (shear strength) of the column is the variable of interest in this case, as it was the distance d_t in the illustrative problem. The data used in this experiment is publicly available and were taken from [41]. In particular, the test No. 1 performed by [52] is used. This data set comprises 626 data points, which are sequential in nature, given that the displacement and shear strength were recorded continuously throughout the loading cycles. The specimen consisted of a double-ended beam column of 3300 [mm] length and 550x550 [mm] cross section (see Figure 8), with 12 reinforcing bars with a nominal diameter of 24 [mm] as longitudinal reinforcement, symmetrically distributed in the cross section. Lateral reinforcement comprised two 10 [mm] diameter stirrups spaced every 80 [mm]. The average concrete compressive strength was measured as 23.1 [MPa]. The yield strength of the longitudinal and transverse reinforcement was 375 [MPa] and 297 [MPa], respectively. The specimen was subjected to a constant axial compressive load of 1815 [kN]. According to [41], the data of the specimen have been adapted to the case of an equivalent cantilever column by means of an equivalent cantilever length. Accordingly, the equivalent length for the selected specimen is set equal to 1200 [mm].

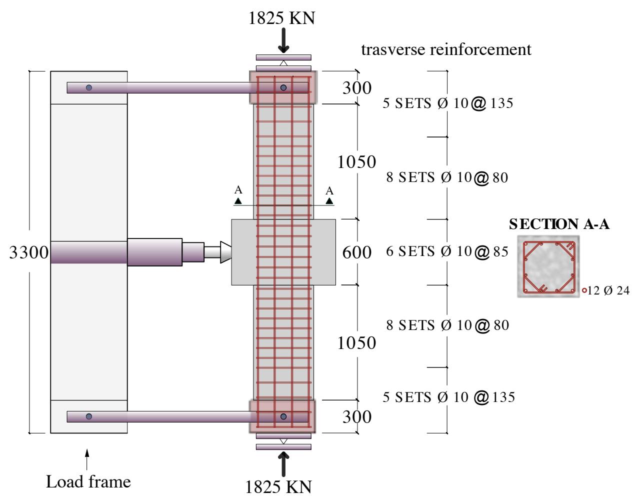


Figure 8: Double-ended reinforced concrete beam-column specimen details, adapted from [52].

The physics-based model used consists of a force-based formulation of a beam-column nonlinear element, fed with fiber sections. This model outputs the shear strength F_m of the column based on: (1) the lateral displacement of the free side of the cantilever, (2) the stiffness and constitutive behavior of the materials, and (3) the geometric characteristics of the element. OpenSeespy software [53] is used to construct the numerical model. The beam-column element deformations are solved using 5 Newton-Cotes integration points, each with the same fiber section. A discretization is done to model the axial and flexure behaviour of the section by means of uni-axial constitutive models, where *Concrete01* and *Steel02* models are used to represent the concrete and steel uni-axial behaviour, respectively. The input parameters of the uni-axial models are defined according to the recommendations given in [54]. Note that the concrete inside the stirrup cage is subjected to lateral pressure due to Poisson's effect and the passive action of the stirrups; and that the lateral pressure affects the uni-axial behaviour of the concrete, giving additional strength and deformation capacity. This behaviour is considered by the confinement factor, which is estimated using the recommendations of [55]. Table 1 summarizes the input data of the numerical model, whereas Figure 9 depicts the configuration of the numerical model and the uni-axial constitutive models of the steel reinforcement and concrete.

The experimental input data fed into the neural networks are three: the lateral displacement d_l, the direction of the displacement d_d (positive or negative), and the number of cycles n_c (where one cycle is a full lateral displacement on each direction). All three inputs are obtained by processing the displacement data in [52]. Therefore, the objective is to predict the lateral force F (shear strength) at a certain time of the experiment given the lateral displacement, the direction of such displacement and the number of cycles that the column has experienced at that point. Moreover, only the first cycles of the experimental data will be used for training, and the rest will be used as test data. Thus, we can evaluate the extrapolation capabilities of the algorithms, based on their ability to make predictions about future cycles.

The input vectors are organised in three-dimensional arrays including lateral displacement, the direction of such displacement, and the number of cycles $[d_l,d_d,n_c]$, while the output vectors are one-dimensional arrays containing the observed force (shear strength) $[F]$. The physics-based information coming from the OpenSeespy model is arranged in one-dimensional arrays $[F_m]$.

Table 1: Input parameters values of the reinforced concrete model in the engineering case study of Section 4.1.2

Axial Force	Steel Yield Strength	Concrete Compressive Strength	Cross Section	Length	Confinement Factor	Longitudinal reinforcement ratio	Strain hardening ratio
1815	375	23.10	550x550	1200	1.70	0.0179	0.0013

4.2. Algorithms and metrics

In this section, the algorithms used in the experiments along with the details about their implementation are presented. Also, a hyperparameter sensitivity analysis is provided.

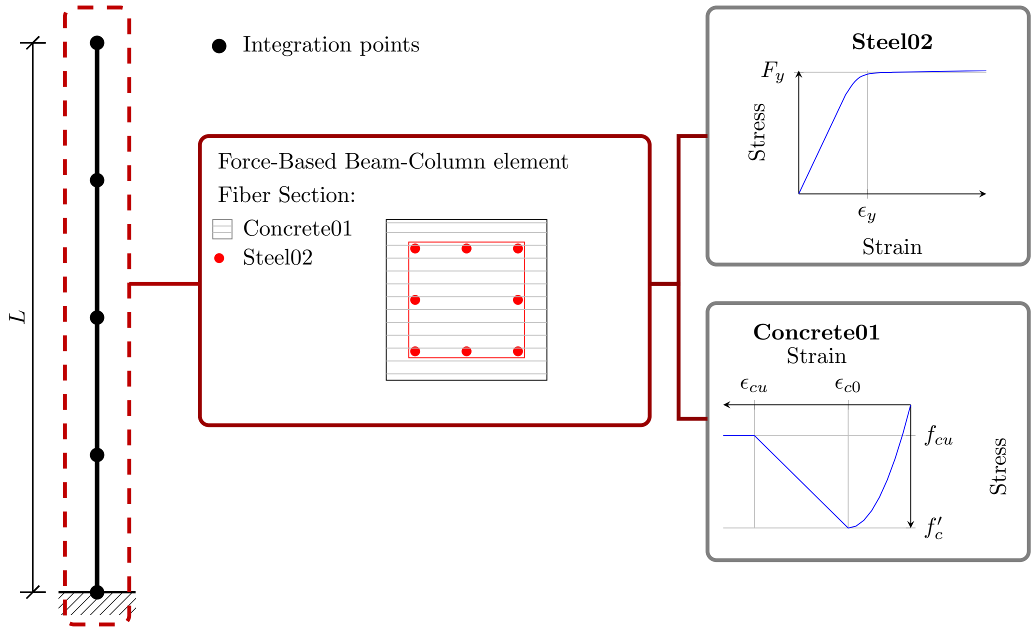


Figure 9: Schematic view of the nonlinear model of a cantilever reinforced concrete beam-column modelled using OpenSeespy. On the right-hand side, plots of the constitutive material monotonic behavior are presented.

4.2.1. Algorithms

The proposed hybrid models have been compared and benchmarked against their data-driven and physics-based counterparts, as well as the state-of-the-art PGNN and a standard ANN, both trained with the backpropagation algorithm using *TensorFlow*, so their performance and potential benefits can be evaluated. The results from this comparison can be found in Tables 2 and 3 and Figures 10-13. The choice of architecture and tuning of the hyperparameters is explained in Section 4.2.2.

The following training algorithms have been used in both experiments. Their architecture comprise two hidden layers with Rectified Linear Units (ReLU) as the activation function, and the output layer with one neuron and a linear activation function. The number of neurons in the input layer varies between the experiments. Note that the physics-guided neural networks, both the proposed models PG-BNN by ABC-SS and the benchmark models state-of-the-art (SOTA) PGNN, have an index number from (1) to (3) depending on where the physics are introduced in the ANN architecture, being (1) through the metric/loss function, (2) through the input neurons, and (3) through the output neurons. Thus, the proposed algorithms can be easily compared against their correspondent state-of-the-art algorithms.

- *BNN by ABC-SS*: A BNN trained with ABC-SS as per Algorithm 1 in [40], to serve as a Bayesian data-driven benchmark. For the illustrative example, the neural network structure comprises two input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters chosen are

P_0=0.1, N=100,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007. In the engineering case study, the same configuration is used but with 3 input neurons and a tolerance value ϵ=0.009.

- Standard ANN with L2 regularization: A standard neural network using *TensorFlow*, to serve as a deterministic data-driven benchmark. *Adam* optimizer [56] with *early stopping* and L2 regularization are used during training. In the illustrative example, the neural network structure comprises two input neurons, 15 neurons per hidden layer, and one output neuron. In the engineering case study, the same configuration is used but with 3 input neurons. The hyperparameters used are L2=0.01, *epochs*=20000 and *patience*=100.

- PbM: Physics-based model to be used as a physics-based benchmark. The model formulation can be found in Equation 5 for the illustrative problem and in Section 4.1.2 for the engineering case study.

- PG-BNN by ABC-SS: The proposed hybrid BNN trained with ABC-SS as per Section 3. Three variants are used as follows:

  - (1): A hybrid BNN as per Section 3.1. For the illustrative problem the neural network structure comprises 2 input neurons, 15 neurons per hidden layer, and one output neuron. The hyperparameters chosen are P_0=0.1, N=100,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007. In the real case study, 3 input neurons and a tolerance value ϵ=0.009 are used. Also, three values of α (0.25, 0.5 and 0.75) have been tested.
  - (2): A hybrid BNN as per Section 3.2. For the illustrative problem the neural network structure comprises 3 input neurons, 5 neurons per hidden layer and one output neuron. The hyperparameters chosen are P_0=0.2, N=10,000, σ_0=0.9, p=0.50 and tolerance value (normalized) ϵ=0.0007. In the real case study, 4 input neurons and a tolerance value ϵ=0.009 are used.
  - (3): A hybrid BNN as per Section 3.3. In this case, the same network structure and hyperparameters as for (2) are used, but with 2 input neurons for the illustrative problem and 3 for the real case study.

- SOTA PGNN: A physics-guided neural network trained with the state-of-the-art backpropagation algorithm using *TensorFlow*, as those described in Section 1, to be used as a physics-guided benchmark. Three variants are tested as follows:

  - (1): A PGNN which follows the present-day approach of introducing the physics in the loss function. The training process uses the backpropagation algorithm to minimize a hybrid loss function, which includes a standard data-driven term ($Loss_d$) and a physics-based one ($Loss_p$), as follows:

$$\arg\min_{(w,b)} Loss_d(\hat{y}, y) + \lambda_p Loss_p(\hat{y}, y_p) \tag{6}$$

where: $Loss_d(\hat{y}, y) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$; $Loss_p(\hat{y}, y_p) = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_{p,n})^2$; $\hat{y}$ is the output of the neural network; y is the training data; and y_p is the output of the physics-based model described in Section 4.1. The neural network architecture is the same as that of *BNN by ABC-SS*, with 15 neurons per hidden layer. *Adam* optimizer [56] with *early stopping* is used for training, and the values of the hyperparameters are λ_p=0.5, *epochs*=20000 and *patience*=100.

  – (2): A PGNN with the architecture presented in Figure 4, where the physics are introduced through the input layer. The number of neurons per layer are the same as PG-BNN by ABC-SS (2). *Adam* optimizer [56] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10000 and *patience*=80.

  – (3): A PGNN with the architecture presented in Figure 6, where the physics are introduced through the output neurons. The number of neurons per layer are the same as PG-BNN by ABC-SS (3). *Adam* optimizer [56] with *early stopping* is used for training, and the values of the hyperparameters are *epochs*=10000 and *patience*=60.

Two different metrics have been chosen to evaluate the performance of the algorithms, taking into account the nature of the tasks and the order of magnitude of the target variables. For the illustrative problem, where the output is the distance d_t in meters [m], root-mean-square-error (RMSE) is used. However, the target variable in the real case scenario is the lateral force F in Newtons [N] which takes significantly large values, therefore, Mean-square-error (MSE) of the normalized data is used.

4.2.2. Sensitivity Analysis

A sensitivity analysis has been undertaken for all algorithms used in the experiments. This study allows us to understand the effect of data and hyperparameters in the overall performance of the models, along with ensuring that the best values of these hyperparameters are chosen. The methodology is explained in this section, the chosen hyperparameters are shown in Section 4.2.1, and the results are presented in Section 4.3. The analysis has been undertaken as follows:

- Data size: In the engineering case study, different ratios of training/test data have been used, namely 20/80, 40/60, 60/40 and 80/20. Thus the performance of the physics-guided and data-driven algorithms under different conditions of availability of data can be evaluated. It can be seen from Table 4, and discussion in Section 4.3.2, that the amount of data used for training has a significant effect on the performance of all algorithms, as could be expected.

- Model architecture: Different architectures have been tested, from multi-layer perceptrons with one single hidden layer, to more complex configurations with 3 hidden layers. The number of units tested per hidden layer varied from 1 to 50. Different validation hold-out sets from the training data were used to identify the best performance with the simplest architecture possible. It was observed that 2 hidden layers provided the best performance with the minimum total number of neurons. Regarding the activation functions, ReLU provided the best results as expected, over others like sigmoid and hyperbolic tangent. With respect to the number of units per hidden layer, 5 neurons per layer were enough to reach good performance in models where the physics are introduced in the forward pass, such as SOTA PGNN (2), SOTA PGNN (3), PG-BNN by ABC-SS (2) and PG-BNN by ABC-SS (3). However, other models that are purely data-driven or the physics are introduced in the loss/metric function, namely Standard ANN, BNN by ABC-SS, SOTA PGNN (1) and PG-BNN by ABC-SS (1), required a slightly higher number of units, 15 per hidden layer, to reach acceptable performance in validation sets. It was observed that beyond those numbers of layers and neurons for each model, more complex architectures with more neurons per hidden layer did not provide a significant improvement in their performance or validation error, but just a slightly increased capacity to overfit the training data. The method to avoid overfitting is described below.

- Model hyperparameters:
  - Training based on backpropagation: The hyperparameters to be tuned are the number of *epochs*, and the *patience* of the early stopping optimizer. Different hold-out data sets within the training set are used as validation, thus the maximum number of epochs required is identified by monitoring the training and validation loss. The *patience* is fixed to a value which avoids overfitting without compromising on model accuracy. Low values of *patience* may lead to underfitting, while higher numbers may stop the training too late, leading to overfitting. The number of *epochs* tested varied from 1000 to 30000, and the *patience* from 1 to 500. In the standard ANN, different values of the the L2 parameter were tested, from 0.001 to 1. Likewise, different values of the hyperparameter λ_p in SOTA PGNN (1) were checked, from 0.1 to 2. Lower values of λ_p means that the physics are not strongly considered, leading to better fit of the model to data, however, higher values penalise data fitting and prioritise the physics, which may improve extrapolation.
  - Training based on ABC-SS: The hyperparameters to be optimised are P_0, N, σ_0, p and ϵ. A similar process was followed, using validation hold-out data sets. In terms of sensitivity, for more complex architectures P_0 needs to be set to a smaller value, while a bigger number of samples N are required. The values of σ_0 and p, which refer to how new samples are drawn from the proposal PDF, are more sensitive and need to be adjusted simultaneously. The value of the tolerance ϵ has a similar effect to the *patience* parameter, as it stops the simulation when a certain error is reached. This value should be set to avoid overfitting, but without compromising the accuracy of the model. The range of values tested for each hyperparameter are as follows: P_0 (0.1, 0.2 and 0.5), N (1000-500000), σ_0 (0.1-2), p (0.1-0.9) and ϵ (0.01-0.0001).

As a final remark, test data sets are not used during training or for hyperparameter tuning, but always reserved for the testing stage only.

4.3. Results and discussion

In this section, the results from the experiments are presented both numerically and graphically. The algorithms and metrics used are those detailed in Section 4.2. A discussion on the results is also included, highlighting the differences found between physics-based models, purely data-driven models and the proposed hybrid models.

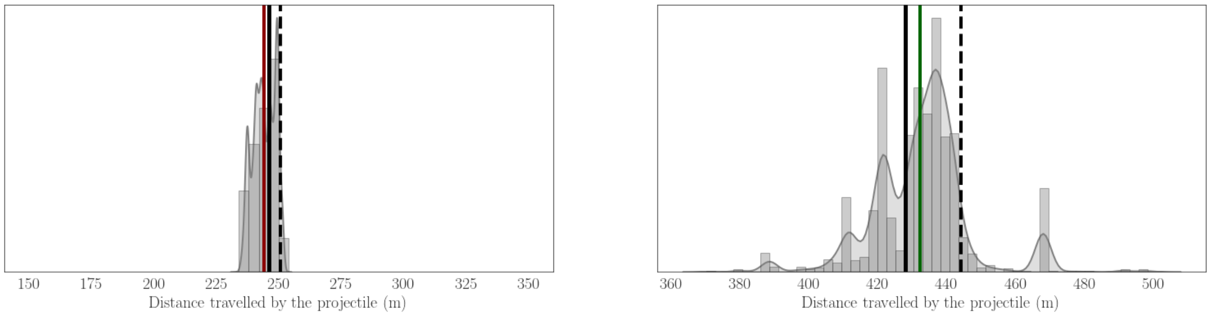
4.3.1. Illustrative problem: projectile motion

As explained in Section 4.1.1, a projectile motion problem is adopted to illustrate the proposed concepts, evaluate the performance of the proposed hybrid algorithms, and compare them against purely data-driven and physics-based models, as well as the SOTA PGNN trained with backpropagation. All algorithms presented in Section 4.2 have been trained and tested with the data sets presented in Section 4.1.1 through 50 independent runs. The RMSE from those runs has been recorded and the results are shown in Table 2. It can be seen that the proposed hybrid models where the laws of physics are introduced in the metric ρ, as per PG-BNN by ABC-SS (1), neither provide better results than the data-driven approach with *BNN by ABC-SS*, nor seem to improve extrapolation. However, the new metric ρ_p may be understood as a regularization tool, which may force the neural network to ignore those training data points that differ significantly from the physics. This suggests that this hybrid model might be useful in those cases where there is a significant amount of noise in the observed data. PG-BNN by ABC-SS (2) has provided better results than the purely data-driven approaches but, even though its predictions on Test Data Set 2 have outperformed those from *BNN by ABC-SS* and Standard ANN, it does not extrapolate better than the physics-based model. The best results are given by PG-BNN by ABC-SS (3) and SOTA PGNN (3), especially when extrapolating in Test Data Set 2, outperforming the physics-based model, the data-driven algorithms, and the the other variants of physics-guided neural networks. The neural network in PG-BNN by ABC-SS (3) seems to find a pattern in the discrepancy between the physics-based model and the observed data which could be, for instance, some environmental conditions not included in the model, like the headwind in our case. Then, when asked to extrapolate, it applies that pattern to the physics included in the overall hybrid model, thus improving the predictions of the purely physics-based model. But most importantly, ABC-SS allows for an accurate quantification of the uncertainty as shown in Figure 10, where the predictions made outside the domain of the training data (extrapolation) are more disperse. That also provides us with valuable information about the degree of belief on the predictions made by the hybrid model. Finally, a mixed model where the physics-based model is introduced in both the input and output neurons was tested, however, it did not provide better results than PG-BNN by ABC-SS (3).

Table 2: Illustrative problem. Comparison between PG-BNN by ABC-SS, *BNN by ABC-SS*, standard ANN, the physics-based model and the state-of-the-art PGNN. The results, expressed in terms of RMSE, were obtained after 50 independent runs of each algorithm.

Statistics of RMSE obtained in 50 independent runs of the training algorithm							
	Neurons per Hidden Layer	Test Data Set 1 (Interpolation)			Test Data Set 2 (Extrapolation)		
		Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})
PG-BNN by ABC-SS (1) (α=0.25)	15	11.618	12.319	13.423	124.756	136.819	149.348
PG-BNN by ABC-SS (1) (α=0.5)	15	10.361	11.288	12.560	116.899	137.623	152.801
PG-BNN by ABC-SS (1) (α=0.75)	15	10.191	11.209	12.294	120.976	131.285	147.468
PG-BNN by ABC-SS (2)	5	5.249	5.909	6.588	21.271	32.211	39.404
PG-BNN by ABC-SS (3)	5	**3.670**	3.856	3.985	**5.253**	**5.919**	**6.833**
BNN by ABC-SS	15	10.254	11.376	12.529	121.682	133.947	146.087
Physics-based Model	N/A	8.780	8.780	8.780	10.527	10.527	10.527
SOTA PGNN (1)	15	23.258	23.945	24.704	113.396	115.289	117.182
SOTA PGNN (2)	5	3.755	**3.766**	**3.788**	31.938	34.780	38.839
SOTA PGNN (3)	5	3.875	3.930	3.967	5.990	6.703	8.407
Standard ANN with L2 Reg	15	5.540	7.905	20.093	126.270	134.994	140.120

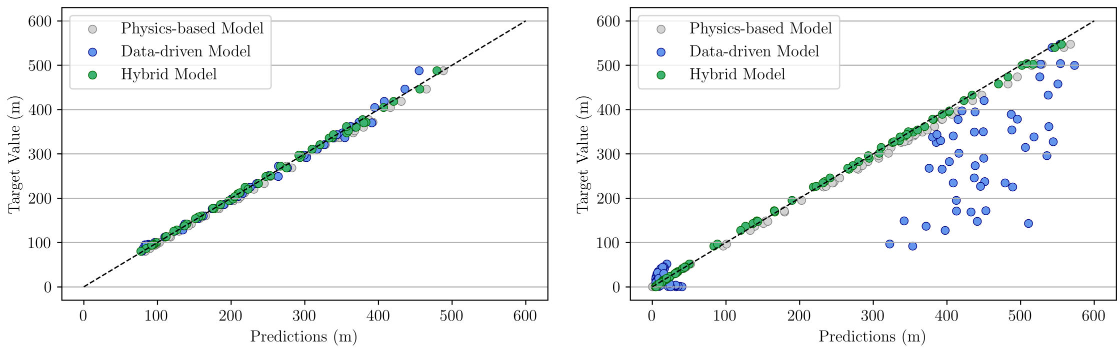
This illustrative experiment has shown that neural networks can help physics-based models to consider complex aspects that were not included in the original model, such as environmental conditions, in the same way that physics-based models can help neural networks to extrapolate outside the domain of the training data. This last aspect is graphically explained in Figure 11, where we see that the hybrid model benefits from both, the data-driven approach to improve the physics-based predictions, and especially from the physics-based model when extrapolating (panel b). That symbiosis brings to light the fact that hybrid models are specially useful when solving engineering problems where data is scarce but there exist relatively simple physics-based models, or at least, some prior knowledge of the task in hand. Also, the use of ABC-SS as learning engine has provided more flexibility and accuracy than standard backpropagation. This Bayesian training is the main advantage of the proposed PG-BNN by ABC-SS over the state-of-the-art methods, as it provides the user with valuable information about the uncertainty present in the observed data. Lastly, it should be noted that the computational cost of the hybrid algorithms in this experiment is comparable to that of their data-driven counterparts. However, if very complex physics-based models with high computational costs are used, then the running time of the hybrid algorithms could be impacted.



(a) Prediction inside the domain of the training data (interpolation)

(b) Prediction outside the domain of the training data (extrapolation)

Figure 10: Illustrative Problem. Probabilistic predictions made by PG-BNN by ABC-SS (3) shown as a light grey density function, within the domain of the training data (interpolation) and outside of it (extrapolation). The mean predictions of the hybrid model are shown in red and green respectively. The predictions made by the purely physics-based model are shown in dashed black line and the true value of the projectile range in continuous black line.



(a) Test Data Set 1 (interpolation)

(b) Test Data Set 2 (extrapolation)

Figure 11: Illustrative problem. Scatter plot of target values against predicted values by the hybrid model PG-BNN by ABC-SS(3) in green, data-driven model *BNN by ABC-SS* in blue and the physics-based model in grey, for Test Data Set 1 (interpolation) in panel (a) and for Test Data Set 2 (extrapolation) in panel (b).

4.3.2. Engineering Case Study: Application to lateral-load tests in reinforced concrete columns

The proposed algorithms have been applied to one of the column tests recorded in the The PEER Structural Performance Database [41] as explained in Section 4.1.2, and benchmarked against the purely data-driven methods, such as *BNN by ABC-SS* and Standard ANN, the physics-based model described in that same section, and the state-of-the-art physics-guided neural networks. The algorithms, along with the choice of architecture and hyperparameters, are explained in Section 4.2 and the results of the experiment can be found in Table 3. Algorithms PG-BNN by ABC-SS (1) and SOTA PGNN (1) are not shown for this experiment given that they do not provide better results, as demonstrated and discussed in Section 4.3.1. Overall, the results of this experiment are similar to those obtained in the illustrative problem. When evaluated on test data, PG-BNN by ABC-SS (3) and SOTA PGNN (3) outperform the other physics-guided neural networks, the physics-based model, *BNN by ABC-SS* and Standard ANN, even when these purely data-driven approaches required a more complex architecture with more neurons in the hidden layers. Once again, the neural network has been able to learn a pattern in the difference between the physics and the data, so when asked to make a prediction about unseen data it compensates the information coming from physics-based model with that pattern, thus it closely matches reality. Also, the time of computation of the hybrid models is significantly lower, given its simpler architecture and relatively small number of samples N required. Interestingly, SOTA PGNN (2) and PG-BNN by ABC-SS (2) achieve low MSE values when evaluated on training data, which may suggest that introducing the physics through the input neurons is more prone to overfitting. This might be because the neural network also manipulates the physics introduced through the input layer to match the observed data. For that same reason, the performance of both SOTA PGNN (2) and PG-BNN by ABC-SS (2) seem to be worse on test data. The quantification of the uncertainty is the main advantage that the proposed hybrid models share with *BNN by ABC-SS*, given that both are trained with approximate Bayesian computation [39, 40]. This is shown in Figure 12, where we see that PG-BNN by ABC-SS (3) not only make better predictions than the physics-based model, especially on test data, but also quantifies the uncertainty realistically. It seems natural that such uncertainty (light grey density function), translated into the width of range of plausible values, grows as we move away from the training data, as in panel (b) of Figure 12 and Figure 13. Lastly, and in line with the results obtained in the illustrative problem, the good performance of PG-BNN by ABC-SS (3) outside the domain of the training data (extrapolation) is notable, as can be seen again in Table 3, Figure 12 panel (b) and Figure 13, where the predictions about future cycles (green line) are acceptably accurate. As a final remark, from the results provided by both PG-BNN by ABC-SS (3) and the physics-guided SOTA PGNN (3) used as benchmark, it may be concluded that introducing the physics through the output neuron provides the best performance. Moreover, PG-BNN by ABC-SS (3) also allows for a flexible quantification of the uncertainty, which will improve the subsequent decision making process. In terms of efficiency, the proposed hybrid models showed comparable running times to that of their data-driven counterparts, as per in the illustrative example.

A sensitivity analysis about the performance of the algorithms based on the availability of data has also been carried out, and the results are shown in Table 4. When data is very scarce, such as 20%, the hybrid models do not seem to benefit from them significantly, as their accuracy on test data is in the same order of magnitude than the purely physics-based model. However, when a greater amount of data is available, such as 40% or 60%, the hybrid models benefit considerably from them and outperform both the data-driven methods and the purely physics-based model. This becomes more evident when 80% of the total data is available for training, as both PG-BNN by ABC-SS (3) and SOTA PGNN (3) provide very accurate predictions in comparison with all other methods.

Regarding the applicability of this experiment to a real world scenario, the seismic structural engineering field could become a good candidate. One of the problems that arise in a post-earthquake scenario is the difficulty in deciding if a structure remains safe and can still be used [57], in relation to the capability of that

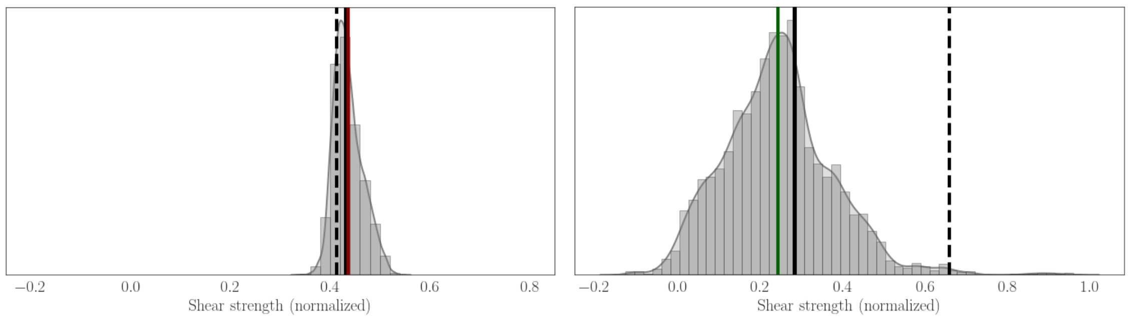
Table 3: Detailed comparison, based on a training/test data ratio of 60/40, between PG-BNN by ABC-SS, *BNN by ABC-SS*, Standard ANN, the purely physics-based model, and the state-of-the-art PGNN. The results, expressed in terms of MSE, were obtained after 50 independent runs of each algorithm.

Statistics of MSE obtained in 50 independent runs of the training algorithm							
	Neurons per Hidden Layer	Training Data Set			Test Data Set		
		Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})	Q1 (P_{25})	Median (P_{50})	Q3 (P_{75})
PG-BNN by ABC-SS (2)	5	0.0042	0.0045	0.0047	0.0103	0.0129	0.0160
PG-BNN by ABC-SS (3)	5	0.0051	0.0054	0.0056	0.0052	**0.0056**	**0.0073**
BNN by ABC-SS	15	0.0050	0.0054	0.0057	0.0157	0.0184	0.0299
Physics-based Model	N/A	0.0308	0.0308	0.0308	0.0521	0.0521	0.0521
SOTA PGNN (2)	5	0.0023	0.0030	0.0038	0.0118	0.0144	0.0243
SOTA PGNN (3)	5	0.0009	0.0011	0.0057	**0.0046**	0.0088	0.0127
Standard ANN with L2 Reg	15	0.0047	0.0052	0.0079	0.0357	0.0494	0.0745

Table 4: Sensitivity analysis about different ratios of training/test data and the accuracy of the algorithms. The results, expressed in terms of MSE, refer to the median value (P_{50}) of the error obtained on test data after 50 independent runs of each algorithm, based on different ratios of training/test data.

Median value (P_{50}) of MSE obtained on test data after 50 independent runs					
	Neurons per Hidden Layer	Percentage of data used for training			
		20%	40%	60%	80%
PG-BNN by ABC-SS (2)	5	0.0392	0.0186	0.0129	0.0112
PG-BNN by ABC-SS (3)	5	0.0460	0.0083	0.0056	0.0031
BNN by ABC-SS	15	0.1947	0.1616	0.0184	0.0162
SOTA PGNN (2)	5	0.0740	0.0540	0.0144	0.0107
SOTA PGNN (3)	5	0.0481	0.0362	0.0088	0.0030
Standard ANN with L2 Reg	15	0.1250	0.1244	0.0494	0.0334
Physics-based Model	NA	0.0459	0.0512	0.0521	0.0587

structure to withstand the aftershocks, all aggravated by the significant uncertainty inherent to this type of phenomena. This can be of special interest for healthcare facilities, where the evacuation (or closure) of the building is not a straightforward decision during an emergency. The combination of visual inspections with the proposed hybrid model framework could become an effective tool for fast evaluation, which is required to take an informed decision during this kind of critical scenarios. Moreover, the presented tool aligns with the current tendency in seismic structural engineering, about the need to account for uncertainties on the behaviour of structural elements [58].



(a) Prediction on training data

(b) Prediction on test data (extrapolation)

Figure 12: Engineering case Study. Mean predictions made by PG-BNN by ABC-SS (3) on training data (red) and test data (green). The uncertainty is represented by the light grey PDF, the prediction of the physics-based model is given by the dashed line and the target value is the black continuous line.

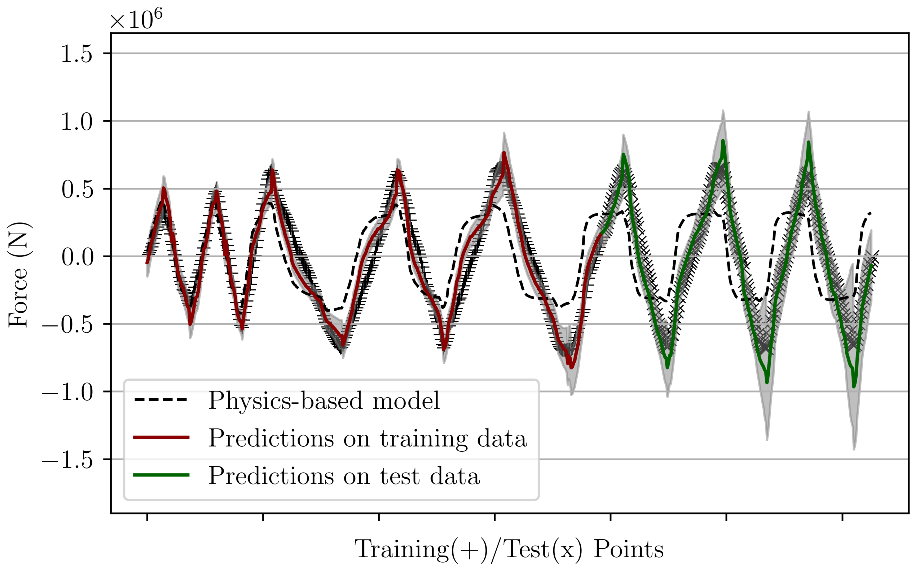


Figure 13: Engineering case study. Predictions about lateral force made by PG-BNN by ABC-SS (3) on training data (red) and on test data (green). The uncertainty is represented by the grey hatch, the prediction of the physics-based model is given by the dashed line, the training data set is represented by + and the test data set is represented by x.

5. Conclusions

This manuscript presented a new algorithm which combines *BNN by ABC-SS* with physics-based models, the so-called PG-BNN by ABC-SS. Unlike other physics-guided/informed neural networks where the physics are often introduced in the loss function or through boundary conditions, and then backpropagated during training, the proposed algorithm inserts the physics directly in the forward pass, which improves the extrapolation capabilities. Moreover, ABC-SS is a Bayesian gradient-free training method that provides the proposed algorithm with stability, flexibility and the ability to quantify the uncertainty. Those properties were evaluated in two experiments, where the accuracy of PG-BNN by ABC-SS (3) was comparable to the benchmark SOTA PGNN (3) trained with backpropagation , and outperformed significantly the performance of the purely physics-based and data-driven approaches.

The two main advantages of PG-BNN by ABC-SS, namely its ability to extrapolate outside the domain of the training data set and to quantify the uncertainty in the predictions, may improve significantly the subsequent decision making process in engineering applications. The results in the engineering case study showed the potential of the proposed algorithm to become, if combined with visual inspections, an effective and fast tool to evaluate and diagnose the condition of structural elements after seismic events. Certainly, a tool that can anticipate the outcome of an event of which there is little data, with a defined degree of confidence, could be particularly useful in different engineering fields. Future research should focus on extending the proposed methodology to other types of artificial neural networks, as well as the application of ABC-SS training to high-dimensional parameter spaces. Also, the use of adaptive activation functions [59–61] should be explored.

Acknowledgements

This paper is part of the ENHAnCE project, which has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 859957. The authors would like to thank Chetan S. Kulkarni at NASA Ames Research Center for his comments and useful discussions, and Mrs. Carmen Salas for her encouragement and support.

References

[1] R. K. Coll, D. Lajium, Modeling and the future of science learning, in: Models and modeling, Springer, 2011, pp. 3–21.

[2] J. Z. Buchwald, R. Fox, The Oxford handbook of the history of physics, OUP Oxford, 2013.

[3] A. White, M. Tolman, H. Thames, H. Withers, K. Mason, M. Transtrum, The limitations of model-based experimental design and parameter estimation in sloppy systems, PLOS Computational Biology 12 (2016).

[4] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2020) 477–508.

[5] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in neural information processing systems 25 (2012) 1097–1105.

[6] R. Vaish, U. Dwivedi, S. Tewari, S. Tripathi, Machine learning applications in power system fault diagnosis: Research advancements and perspectives, Engineering Applications of Artificial Intelligence 106 (2021) 104504.

[7] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Transactions on knowledge and data engineering 21 (9) (2009) 1263–1284.

[8] P. J. Haley, D. Soloway, Extrapolation limitations of multilayer feedforward neural networks, in: [Proceedings 1992] IJCNN International Joint Conference on Neural Networks, Vol. 4, IEEE, 1992, pp. 25–30.

[9] 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, Journal of Computational Physics 378 (2019) 686–707.

[10] G. Pang, L. Lu, G. E. Karniadakis, fpinns: Fractional physics-informed neural networks, SIAM Journal on Scientific Computing 41 (4) (2019) A2603–A2626.

[11] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering 365 (2020) 113028.

[12] A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations., in: AAAI Spring Symposium: MLPS, 2021.

[13] N. Zobeiry, K. D. Humfeld, A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications, Engineering Applications of Artificial Intelligence 101 (2021) 104232.

[14] F. Arnold, R. King, State–space modeling for control based on physics-informed neural networks, Engineering Applications of Artificial Intelligence 101 (2021) 104195.
[15] H. Guo, X. Zhuang, T. Rabczuk, A deep collocation method for the bending analysis of kirchhoff plate, arXiv preprint arXiv:2102.02617 (2021).
[16] M. A. Nabian, R. J. Gladstone, H. Meidani, Efficient training of physics-informed neural networks via importance sampling, Computer-Aided Civil and Infrastructure Engineering (2021).
[17] H. Sun, L. Peng, J. Lin, S. Wang, W. Zhao, S. Huang, Microcrack defect quantification using a focusing high-order sh guided wave emat: the physics-informed deep neural network guwnet, IEEE Transactions on Industrial Informatics 18 (5) (2021) 3235–3247.
[18] H. Sun, L. Peng, S. Huang, S. Li, Y. Long, S. Wang, W. Zhao, Development of a physics-informed doubly fed cross-residual deep neural network for high-precision magnetic flux leakage defect size estimation, IEEE Transactions on Industrial Informatics 18 (3) (2021) 1629–1640.
[19] S. Mishra, R. Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for pdes, IMA Journal of Numerical Analysis 42 (2) (2022) 981–1022.
[20] Z. Hu, A. D. Jagtap, G. E. Karniadakis, K. Kawaguchi, When do extended physics-informed neural networks (xpinns) improve generalization?, arXiv preprint arXiv:2109.09444 (2021).
[21] R. Stewart, S. Ermon, Label-free supervision of neural networks with physics and domain knowledge, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[22] R. G. Nascimento, M. Corbetta, C. S. Kulkarni, F. A. Viana, Hybrid physics-informed neural networks for lithium-ion battery modeling and prognosis, Journal of Power Sources 513 (2021) 230526.
[23] R. G. Nascimento, F. A. Viana, Cumulative damage modeling with recurrent neural networks, AIAA Journal 58 (12) (2020) 5459–5471.
[24] R. G. Nascimento, F. A. Viana, Fleet prognosis with physics-informed recurrent neural networks, arXiv preprint arXiv:1901.05512 (2019).
[25] W. De Groote, E. Kikken, E. Hostens, S. Van Hoecke, G. Crevecoeur, Neural network augmented physics models for systems with partially unknown dynamics: Application to slider-crank mechanism, IEEE/ASME Transactions on Mechatronics (2021).
[26] J. Wang, Y. Li, R. Zhao, R. X. Gao, Physics guided neural network for machining tool wear prediction, Journal of Manufacturing Systems 57 (2020) 298–310.
[27] R. Zhang, Y. Liu, H. Sun, Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling, Engineering Structures 215 (2020) 110704.
[28] U. Inyang-Udoh, S. Mishra, A physics-guided neural network dynamical model for droplet-based additive manufacturing, IEEE Transactions on Control Systems Technology (2021).
[29] A. Karpatne, W. Watkins, J. Read, V. Kumar, Physics-guided neural networks (pgnn): An application in lake temperature modeling, arXiv preprint arXiv:1710.11431 (2017).
[30] X. Jia, J. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, V. Kumar, Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles, in: Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, 2019, pp. 558–566.
[31] L. Yang, X. Meng, G. E. Karniadakis, B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data, Journal of Computational Physics 425 (2021) 109913.
[32] D. Zhang, L. Lu, L. Guo, G. E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, Journal of Computational Physics 397 (2019) 108850.
[33] F. N. Jabir B, Dropout, a basic and effective regularization method for a deep learning model: a case study, Indonesian Journal of Electrical Engineering and Computer Science 24 (2) (2021) 1009–1016.
[34] A. Daw, R. Q. Thomas, C. C. Carey, J. S. Read, A. P. Appling, A. Karpatne, Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling, in: Proceedings of the 2020 siam international conference on data mining, SIAM, 2020, pp. 532–540.
[35] Z. Ghahramani, Probabilistic machine learning and artificial intelligence, Nature 521 (7553) (2015) 452–459.
[36] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536.
[37] L. Lu, Dying relu and initialization: Theory and numerical examples, Communications in Computational Physics 28 (5) (2020) 1671–1706.
[38] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International conference on machine learning, PMLR, 2013, pp. 1310–1318.
[39] M. Chiachio, J. L. Beck, J. Chiachio, G. Rus, Approximate Bayesian computation by subset simulation, SIAM journal on scientific computing (3) (2014) A1339—-A1358.
[40] J. Fernández, M. Chiachío, J. Chiachío, R. Muñoz, F. Herrera, Uncertainty quantification in neural networks by approximate bayesian computation: Application to fatigue in composite materials, Engineering Applications of Artificial Intelligence 107 (2022) 104511.
[41] M. Berry, M. Parrish, M. Eberhard, Peer structural performance database user's manual (version 1.0), University of California, Berkeley (2004).
[42] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale

machine learning on heterogeneous systems, software available from tensorflow.org (2015).
[43] U. Hasson, S. A. Nastase, A. Goldstein, Direct fit to nature: an evolutionary perspective on biological and artificial neural networks, Neuron 105 (3) (2020) 416–434.
[44] O. Fink, Q. Wang, M. Svensen, P. Dersin, W.-J. Lee, M. Ducoffe, Potential, challenges and future directions for deep learning in prognostics and health management applications, Engineering Applications of Artificial Intelligence 92 (2020) 103678.
[45] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural network, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, pp. 1613–1622.
[46] M. Betancourt, A conceptual introduction to hamiltonian monte carlo, arXiv preprint arXiv:1701.02434 (2017).
[47] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A survey, arXiv preprint arXiv:2003.04919 1 (1) (2020) 1–34.
[48] R. Rai, C. K. Sahu, Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus, IEEE Access 8 (2020) 71050–71073.
[49] M. Ahmadi, M. Khashei, Current status of hybrid structures in wind forecasting, Engineering Applications of Artificial Intelligence 99 (2021) 104133.
[50] M. A. Chao, C. Kulkarni, K. Goebel, O. Fink, Fusing physics-based and deep learning models for prognostics, Reliability Engineering & System Safety 217 (2022) 107961.
[51] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometrics and intelligent laboratory systems 2 (1-3) (1987) 37–52.
[52] W. Gill, Ductility of rectangular reinforced concrete columns with axial load, Ph.D. thesis, University of Canterbury (1979).
[53] M. Zhu, F. McKenna, M. H. Scott, Openseespy: Python library for the opensees finite element framework, SoftwareX 7 (2018) 6–11.
URL `https://doi.org/10.1016/j.softx.2017.10.009`
[54] M. M. Karthik, J. B. Mander, Stress-block parameters for unconfined and confined concrete based on a unified stress-strain model, Journal of Structural Engineering (2011) 270–273.
URL `https://doi.org/10.1061/(ASCE)ST.1943-541X.0000294`
[55] J. B. Mander, N. Priestley, R. Park, Theoretical stress-strain model for confined concrete, Journal of structural engineering (1988).
URL `https://doi.org/10.1061/(ASCE)0733-9445(1988)114:8(1804)`
[56] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
[57] FEMA, FEMA P-58-1: Seismic performance assessment of buildings. Volume 1–methodology, Vol. 10, Federal Emergency Management Agency, 2012.
[58] E. A. Opabola, K. J. Elwood, Collapse performance of nominally identical nonductile circular columns susceptible to failure-mode variability, Journal of Structural Engineering 147 (6) (2021) 04021069.
[59] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, Journal of Computational Physics 404 (2020) 109136.
[60] A. D. Jagtap, K. Kawaguchi, G. Em Karniadakis, Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks, Proceedings of the Royal Society A 476 (2239) (2020) 20200334.
[61] A. D. Jagtap, Y. Shin, K. Kawaguchi, G. E. Karniadakis, Deep kronecker neural networks: A general framework for neural networks with adaptive activation functions, Neurocomputing 468 (2022) 165–180.