Computers and Structures 245 (2021) 106458

Contents lists available at ScienceDirect

Computers and Structures

journal homepage: www.elsevier.com/locate/compstruc

Estimating model inadequacy in ordinary differential equations with physics-informed neural networks

Felipe A.C. Viana *, Renato G. Nascimento, Arinan Dourado, Yigit A. Yucesan

Department of Mechanical and Aerospace Engineering, University of Central Florida, Orlando, FL 32816-8030, USA

ARTICLE INFO

Article history:
Received 9 August 2020
Accepted 17 November 2020

Keywords:
Physics-informed machine learning
Scientific machine learning
Uncertainty quantification
Recurrent neural networks
Directed graph models

ABSTRACT

A number of physical systems can be described by ordinary differential equations. When physics is well understood, the time dependent responses are easily obtained numerically. The particular numerical method used for integration depends on the application. Unfortunately, when physics is not fully understood, the discrepancies between predictions and observed responses can be large and unacceptable. In this paper, we propose an approach that uses observed data to estimate the missing physics in the original model (i.e., model-form uncertainty). In our approach, we first design recurrent neural networks to perform numerical integration of the ordinary differential equations. Then, we implement the recurrent neural network as a directed graph. This way, the nodes in the graph represent the physics-informed kernels found in the ordinary differential equations. We quantify the missing physics by carefully introducing data-driven in the directed graph. This allows us to estimate the missing physics (discrepancy term) even for hidden nodes of the graph. We studied the performance of our proposed approach with the aid of three case studies (fatigue crack growth, corrosion-fatigue crack growth, and bearing fatigue) and state-of-the-art machine learning software packages. Our results demonstrate the ability to perform estimation of discrepancy, reducing gap between predictions and observations, at reasonable computational cost.

1. Introduction

While physics-based models are built using the best understanding of a given phenomenon, limitations in terms of computational cost and even knowledge of physics often impose limitations to the predictive ability of such models [1,2]. Under such limitations, model predictions do not always agree with observations, even after model parameters are calibrated [3]. Therefore, one can appreciate the importance of research in the field of uncertainty quantification. While most popular approaches lean towards statistical learning methods [4,5], there has been growing interest in modern deep learning approaches, specially physics-informed neural networks [6].

As opposed to the popular use of neural networks, physics-informed neural networks operate on top of governing differential equations. Instead of only reducing the mismatch between neural network predictions and observed data, the training of physics-informed neural networks uses the laws of physics to help handling the reduced number of data points, and constrain the hyper-parameter space (see [7] for further reference). With incompressible fluids, for example, this is achieved by discarding non-realistic solutions violating the conservation of mass principle. Raissi [8] approximated the unknown of solution of partial differential equations by two deep neural networks. The first network acts as a prior on the unknown solution (enabling to avoid ill-conditioned and unstable numerical differentiation). The second network works as a fine approximation to the spatiotemporal solution. The methodology was tested on a variety of equations used in fluid mechanics, nonlinear acoustics, gas dynamics, and other fields. Wu et al. [9] discussed, in depth, how to augment turbulence models with physics-informed machine learning. They demonstrated a procedure for computing mean flow features based on the integrity basis for mean flow tensors and propose using machine learning to predict the linear and nonlinear parts of the Reynolds stress tensor separately. They used the flow in a square duct and the flow over periodic hills to evaluate the performance of the proposed method. Hesthaven and Ubbiali [10] proposed a non-intrusive reduced-basis method (using proper orthogonal decomposition and neural networks) for parametrized steady-state partial differential equations. The method extracts a reduced basis from a collection of snapshots through proper orthogonal decomposition and employs multilayer perceptrons to approximate the coefficients of the reduced model. They successfully tested the proposed method on the nonlinear Poisson equation in

* Corresponding author.
E-mail address: viana@ucf.edu (F.A.C. Viana).

https://doi.org/10.1016/j.compstruc.2020.106458
0045-7949/

one and two spatial dimensions, and on two-dimensional cavity viscous flows, modeled through the steady incompressible Navier–Stokes equations. Swischuk et al. [11] demonstrated through case studies (predictions of the flow around an airfoil and structural response of a composite panel) that proper orthogonal decomposition is an effective way to parametrize a high-dimensional output quantity of interest in order to define a low-dimensional map suitable for data-driven learning. They tested a variety of machine learning methods such as artificial neural networks, multivariate polynomial regression, k-nearest neighbor, and decision trees. Samaniego et al. [12] presented an approach for discretization of partial differential equations using deep neural networks. Authors discuss similarities with collocation methods and how energy functionals found in many mechanics applications lend themselves into loss functions that can be used for training the deep neural networks. They presented numerical results for cases in linear elasticity, elastodynamics, hyperelasticity, phase field modeling of fracture, and piezoelectricity. Goswami et al. [13] studied the usage of physics-informed neural networks to solve brittle fracture problems. Authors propose the hybrid framework to reformulate the crack path prediction problem, where deep neural networks are guided through boundary conditions, and a loss function based on the variational energy. They showcase their approach on six different fracture mechanics problems, and proved the accuracy as well as the efficiency of the proposed method.

Research on physics-informed machine learning has been concentrated on solving differential equations and/or deriving them with deep neural networks. While these are important research thrusts, many engineering applications often impose limited and/or poor data sets and physics that is only partially understood. For example, in diagnosis and prognosis of industrial equipment (e.g., wind turbines, jet engines, airplanes, etc.), predictive models for estimating hardware distress are used to reduce unscheduled maintenance and downtime [14]. In the three examples shown in this paper, operational data is limited and contains mostly controller data; structural health monitoring systems are not always available; inspection data is limited; and complex failure mechanisms are partially understood. Therefore, physics alone would fail to accurately predict and forecast hardware distress while machine learning alone would struggle with poor data sets.

In this work, instead of using machine learning to replace the physics or fit the data, we will focus on how to explicitly code the physics-based domain knowledge into deep neural networks while using machine learning to quantify and compensate for discrepancy with respect to observations. We propose introducing a model discrepancy term into a given ordinary differential equation in the form of a neural network. The neural network parameters are then estimated by minimizing the difference between few observed outputs and the hybrid model predictions. With that, engineers and scientists will be able to use physics-informed layers to model relatively well understood phenomenon (e.g., fatigue crack growth) and neural network layers to assist modeling poorly characterized phenomenon (e.g., transfer function between sensor data and internal loads).

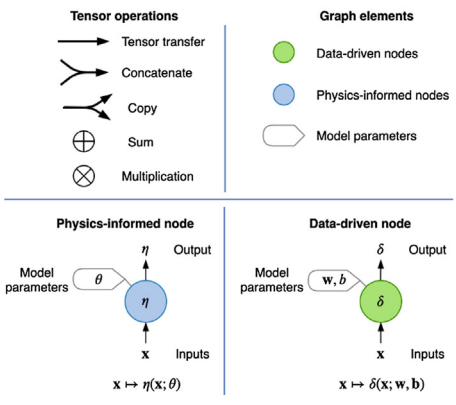
The remaining of the paper is organized as follows. Section 2 presents an overview of direct graph models and neural networks (foundation for solving ordinary differential equation with deep neural networks). Section 3 presents our framework for physics-informed neural networks and how to use graph models for estimation of missing physics. Section 4 presents and describes the numerical experiments and presents the results along with discussion. Finally, Section 5 closes the paper recapitulating salient points and presenting concluding remarks.

2. Neural networks overview

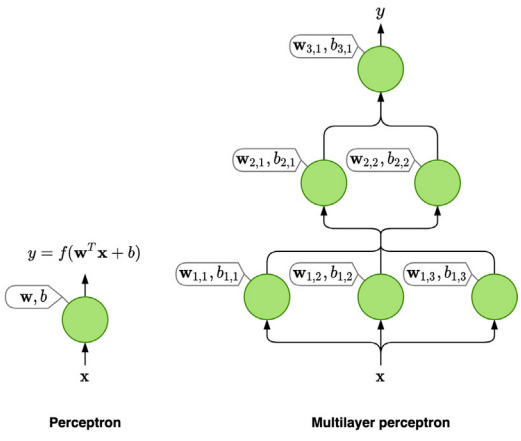
2.1. Neural networks as directed graph models

Fig. 1a introduces the notation and elements we used in this paper. Tensors are used to represent inputs and outputs. The basic tensor operators include tensor transfer (which takes the tensor from one node of the graph to another), concatenation, copy, as well as algebraic operators, such as sum and multiplication. Nodes implement complex operations taking tensors as inputs and returning tensors as outputs. As we will describe later, nodes can implement physics-informed models. Traditionally, in neural networks, nodes implement data-driven models. As shown in Fig. 1b, a directed graph consists of finitely many nodes and edges, with edges directed from one node to another [15,16]. We can use the idea to represent popular neural network architectures such as the perceptron or multilayer perceptron.

As we will detail in Section 3, we propose directly implementing ordinary differential equations that describe the physics of a problem as deep neural networks using directed graph models. Therefore, optimization of hyperparameters is done through back-propagation. As illustrated in Fig. 2, there are essentially two steps in every iteration of the optimization algorithm. First, training data is fed forward, generating the corresponding outputs (Fig. 2a), prediction error, and finally the loss function. Then, the loss function adjoint is propagated backward (through the chain rule) giving

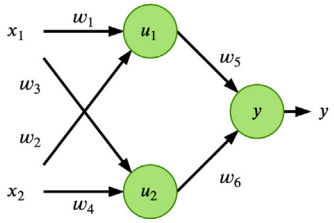


(a) Notation.

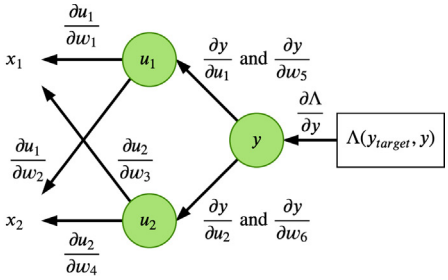


(b) Perceptron and multilayer perceptron, $f(.)$ is an activation function.

Fig. 1. Graph representation of neural networks.



(a) Prediction (forward pass).



(b) Training (backward pass).

Fig. 2. Backpropagation overview. In prediction, inputs are fed forward, generating the activations of hidden layers u_i and output layer y. In training, partial derivatives are fed backward, generating the gradient of the loss function with respect to the weights, $\nabla\Lambda = \left[\frac{\partial\Lambda}{\partial w_1} \dots \frac{\partial\Lambda}{\partial w_6}\right]^T$.

the gradient with respect to the parameters to be optimized. Fig. 2b is a graphical representation of backward pass for the multilayer perceptron. Even though the figure only shows the weight hyper-parameters ($\mathbf{w}$), the formulation can be extended for the case where each perceptron also has a bias term (b). Formally, from Fig. 2b, the gradient of the loss function with respect to the weights can be written as

$$\begin{aligned} &\frac{\partial\Lambda}{\partial w_1} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial u_1}\frac{\partial u_1}{\partial w_1}, \quad \frac{\partial\Lambda}{\partial w_2} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial u_1}\frac{\partial u_1}{\partial w_2}, \quad \frac{\partial\Lambda}{\partial w_3} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial u_2}\frac{\partial u_2}{\partial w_3}, \\ &\frac{\partial\Lambda}{\partial w_4} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial u_2}\frac{\partial u_2}{\partial w_4}, \quad \frac{\partial\Lambda}{\partial w_5} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial w_5}, \text{ and } \quad \frac{\partial\Lambda}{\partial w_6} = \frac{\partial\Lambda}{\partial y}\frac{\partial y}{\partial w_6}. \end{aligned} \tag{1}$$

Fig. 2 also gives insight into two important aspects of our framework. Second, in the backward pass, it is important to have the adjoints readily available. With that in place, one can start implementing the ordinary differential equations as deep neural networks.

2.2. Recurrent neural networks

Recurrent neural networks are specially suitable for dynamical systems [17–19]. They extend traditional feed forward networks to handle time-dependent responses, as shown in Fig. 3a. Recurrent neural networks have been used to model time-series [20], speech recognition [21], diagnosis and prognosis [22–25], and many other applications. A recurrent neural network [17] repeatedly apply transformations to given states in a sequence such that

$$\mathbf{y}_t = f(\mathbf{x}_t, \mathbf{y}_{t-1}), \tag{2}$$

where $t \in [0, \dots, T]$ represent the time discretization, $\mathbf{y} \in R^{n_y}$ are the states representing the sequence, $\mathbf{x} \in R^{n_x}$ are input variables, and $f(.)$ defines the transition between time steps (function of input variables and previous states). In the recurrent neural network terminology, different implementations of $f(.)$ are referred to as cells.

Sophisticated recurrent neural network cells have been proposed to overcome training difficulties associated with "vanishing or exploding gradients" [17]. Fig. 3b illustrates two well-known designs, the long short-term memory [26,27] and the gated recurrent unit [28] cells. These cells exhibit complex implementations, with inputs, outputs (and even auxiliary states) passing through multiple non-linear nodes within the cell.

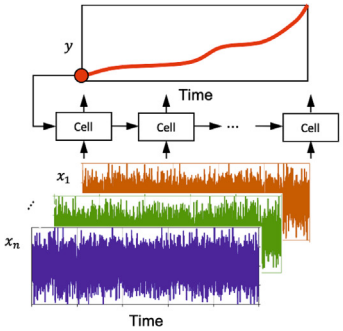
3. Compensating for missing physics in ordinary differential equations

The approach we propose in this paper can be used for any ordinary differential equation. Nevertheless, for simplicity, we will detail the formulation and illustrate case studies of systems described by an initial value problem

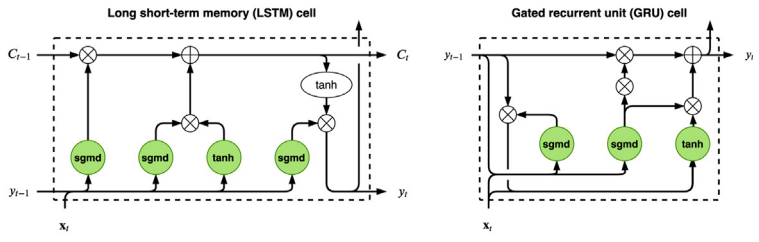
$$\frac{dy}{dt} = f(\mathbf{x}(t), y), \tag{3}$$

such that $\mathbf{x}(0) = \mathbf{x}_0$ and $y(0) = y_0$ are known.

In engineering applications, the answer of Eq. (3) is usually obtained through numerical integration (e.g., Riemann, Euler, Runge-Kutta, etc. [29]). In this work, we propose using the knowledge about the application at hand and the chosen numerical integration method to define the implementation of Eq. (3) as deep neural network. For example, if the application is such that the input time series $\mathbf{x}(t)$ is known through time and the function $f(\mathbf{x}(t), y)$ can be easily evaluated, then we can simply use the Euler's forward method (normalizing to a fixed unit time step) to obtain



(a) Basic idea.



(b) LSTM and GRU cells.

Fig. 3. Recurrent neural network. Cells repeatedly apply transformation to outputs, y_t, and inputs, $\mathbf{x}_t$. Green "sgmd" and "tanh" circles are perceptrons with sigmoid and tanh activations. White "tanh" oval simply applies the tanh activation.

$$y_n = y_0 + \sum_{t=1}^{n} f(\mathbf{x}_t, y_{t-1}). \tag{4}$$

As illustrated in Fig. 4, designing a recurrent neural network cell that performs Euler integration is straightforward. Fig. 4a shows that, in its most general form, all we need to specify is a model for $f(\mathbf{x}_t, y_{t-1})$. Interestingly, if we only wanted to leverage the ability to perform numerical integration, we could implement recurrent neural networks that are purely physics-informed. For example, Fig. 4b illustrates the recurrent neural network cell used in the Euler integration of an initial value problem for which the response increment, $f(\mathbf{x}_t, y_{t-1})$, used in Eq. (4) is defined as

$$f(\mathbf{x}_t, y_{t-1}) = \eta_2(\eta_1(\mathbf{x}_t, y_{t-1})). \tag{5}$$

Nevertheless, in this paper, we argue that a more interesting usage of the Euler recurrent neural network cell is in estimation of model-form uncertainty associated with physics-informed implementations. In engineering applications, both $\eta_1(.)$ and $\eta_2(.)$ carry a physical meaning. Assume that after engineering analysis, $\eta_1(.)$ is known to only partially capture its expected behavior. If that is the case, one can compensate for the missing physics in $\eta_1(.)$ by adding a correction term $\delta(\mathbf{x}_t, y_{t-1})$ before passing it to $\eta_2(.)$

$$f(\mathbf{x}_t, y_{t-1}) = \eta_2(\eta_1(\mathbf{x}_t, y_{t-1}) + \delta(\mathbf{x}_t, y_{t-1})). \tag{6}$$

Fig. 4c illustrates the Euler recurrent neural network cell implementation of Eq. (6). It is worth mentioning that while $\eta_1(.)$ and $\eta_2(.)$ are physics-informed nodes, $\delta(.)$ is a data-driven node that works as discrepancy corrector model. Depending on the application, $\delta(.)$ can assume the form of a multilayer perceptron, a convolutional network, an autoencoder, etc. This way, on top of being a flexible modeling approach, the training of this hybrid physics-informed neural network does not require tedious operations such as inversion of large matrices, computing distance between points, etc. Similarly to any other neural network, the training can rely on well established backpropagation for hyper-parameter optimization.

In our proposed framework, we advocate for the use of physics-informed nodes (i.e., the set of $\eta_i(.)$ in Fig. 4b and c) that are cheap to compute. Computationally efficient nodes are important as these graph models are called many times both during training as well as in prediction (applications including uncertainty quantification, design optimization, inverse problems, model identification, etc.). We know that not all realistic problems in science and engineering lead to cheap-to-evaluate physics-informed nodes. In such cases, we recommend using projection-based techniques to improve computational efficiency when possible [30,31,11].

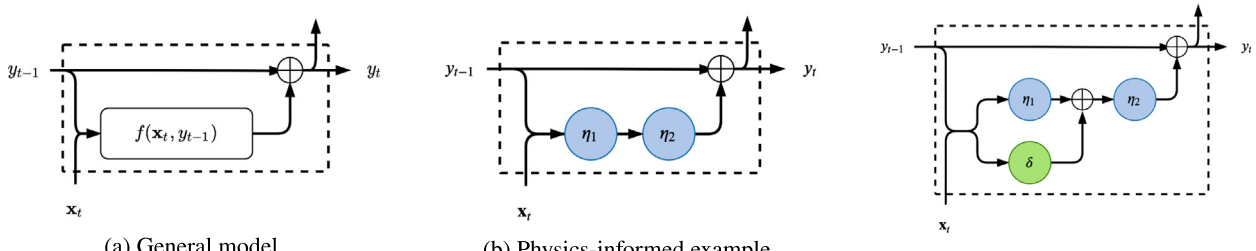
Furthermore, although less frequently, we foresee cases in which the physics-informed nodes have parameters to be optimized. These parameters can be directly incorporated in the backpropagation-based optimization as long as the gradient of the node output with respect to these parameters is available. We recognize that not all implementations of physics-informed nodes would have this gradient information readily available. For the cases in which the gradients can not be implemented directly, we do not favor the implementation through finite differences and alike, as these tend to have undesirable errors and do not scale well with dimensionality. Alternatively, we recommend implementation considering automatic differentiation (an accessible review on automatic differentiation is found in [32]). Thankfully, most modern computational libraries and frameworks for deep learning [33–35] have inherent automatic differentiation capabilities to help users during implementation of customized architectures.

As we will demonstrate in Section 4, the niche of applications where we see our proposed approach outperform traditional data-driven architectures (such as LSTM and GRU shown in Fig. 3b) is when there is huge unbalance in the available data. As illustrated in Fig. 5a, when inputs and outputs are observed throughout the time history, we believe that both purely data-driven and hybrid physics-informed models would perform equally well. However, as shown in Fig. 5b, some problems are characterized by very long sequences (thousands of time steps) with inputs observed throughout time and output observed only at few specific time steps. Under such circumstances, data-driven models might not have a chance to learn the expected input-output behavior. On the other hand, the hybrid physics-informed models are expected to filter out unfeasible/unexpected solutions during training. That way, Fig. 5c illustrates the possible predictions out of these two classes of models.

4. Numerical experiments and discussion

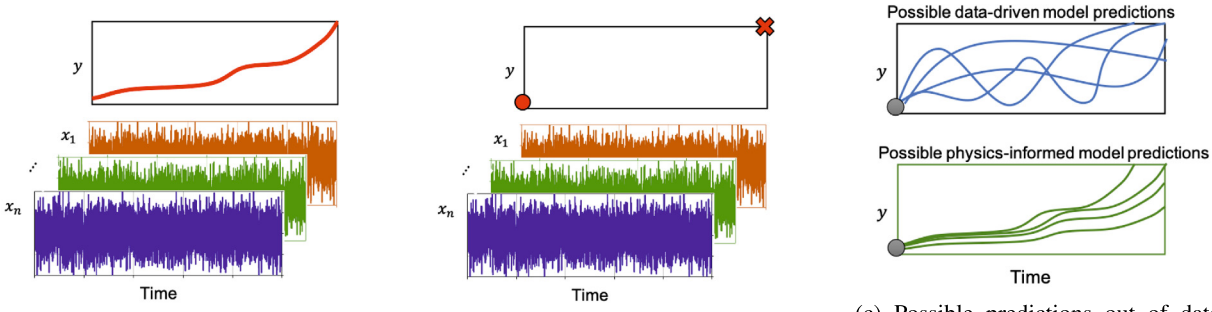
In this section, we will present three different case studies. Each one was designed to test and highlight the ability of our proposed framework to quantify missing physics in systems modeled through ordinary differential equations. The three case studies are:

1. Fatigue of aircraft fuselage panels: Paris law used to model fatigue crack growth. The Euler recurrent neural network is modeled through two nodes in series and uncertainty in the first node is compensated by replacing physics-informed node with a data-driven node.
2. Corrosion-fatigue in aircraft wing: Walker's equation used to model corrosion-fatigue crack growth. The Euler recurrent neural network is modeled through two physics-informed nodes in series accounting for mechanical fatigue contribution and one data-driven node in parallel accounting for contribution due to corrosion.
3. Wind turbine bearing fatigue: Palmer-Miner's rule used to model bearing fatigue. While physics-informed nodes model different aspects of bearing fatigue, a data-driven node is used to model grease degradation (which is a strong contributor to bearing fatigue).



(a) General model. (b) Physics-informed example. (c) Hybrid example.

Fig. 4. Euler integration recurrent neural network cell.



(a) Example of data needed in data-driven applications.

(b) Example of data available for physics-informed applications.

(c) Possible predictions out of data-driven and physics-informed models fitted with data shown in Fig. 5b.

Fig. 5. Data requirements and possible predictions of data-driven and physics-informed models.

4.1. Fatigue of aircraft fuselage panels

In this use case, we consider a control point on a fuselage panel that is inspected in regular intervals through non-destructive evaluation approaches (e.g., Eddy current [36], ultrasound [37], dye penetrant inspection [38], etc.). Scheduled inspection is adopted to reduce cost associated with inspection (mainly downtime, parts, and labor). With large fleets of aircraft, it is common to inspect few aircraft at a time to avoid grounding the entire fleet. As inspection data is gathered, the predictive models used for fleet management are updated. In turn, the updated models can be used to guide the decision of which aircraft should be inspected next.

Consider the control point on an airplane fuselage illustrated in Fig. 6a (crack in infinite plate) and assume that fatigue damage accumulates throughout its useful life. Fatigue crack propagation is a function of time-dependent loads and is usually modeled through Paris's law [39]

$$\frac{da}{dt} = C[\Delta K(t)]^m, \tag{7}$$

where a is the fatigue crack length, C and m are material properties, $\Delta K(t)$ is the stress intensity range. For the sake of this example, assume that the aluminium alloy is characterized by $C = 1.5 \times 10^{-11}$ and $m = 3.8$ and that the initial and the maximum allowable crack lengths are $a_0 = 0.005$ m and $a_{max} = 0.05$ m, respectively.

If the fuselage panel is modeled as a plate with load applied perpendicularly to the crack plane, then the stress intensity range is given by [40,41]:

$$\Delta K(t) = F\Delta S(t)\sqrt{\pi a(t)}, \tag{8}$$

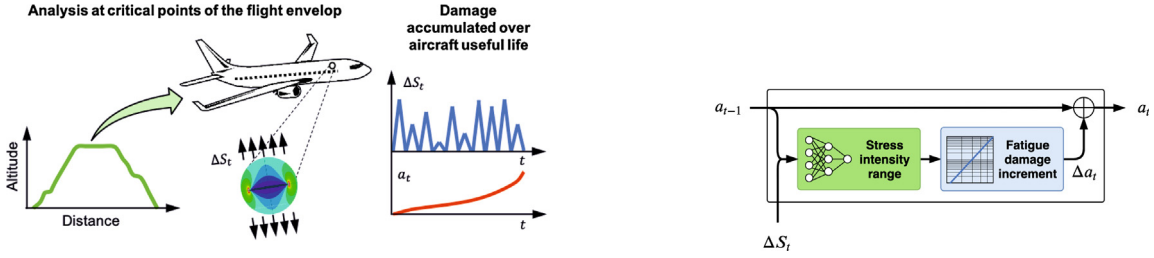
where $\Delta S(t)$ is the far-field cyclic stress time history and F is a dimensionless function of geometry.

In the absence of dedicated structural health monitoring systems and assuming that operating conditions (e.g., Mach number, aircraft attitude, pressures, etc.) are available throughout time, one can use engineering models to estimate $\Delta S(t)$ (high-fidelity finite element analysis together with cycle count methods such as the rain flow method [42]). This way, the integration of Eq. (7) should be straightforward. Unfortunately, the far-field stresses are inevitably limited by the engineering assumptions and simplifications used to build these models, and as a result, $\Delta K(t)$ suffers from model-form uncertainty. In this paper, we overcome that limitation by replacing the physics-informed node that would estimate $\Delta K(t)$ using Eq. (8) with a data-driven implementation. This is convenient since $\Delta K(t)$ is a function of only $\Delta S(t)$ and $a(t)$; while $\Delta S(t)$ is a complex function of operating conditions and the load path to the control point. By replacing $\Delta K(t)$ with a data-driven node, we will be able to compensate for the model form uncertainty originated from $\Delta S(t)$ and adjust $\Delta K(t)$ so that the gap between predicted and observed crack length is reduced. We recognize that one could be interested in adjusting $\Delta S(t)$. We believe that could be possible although potentially more challenging than simply adjusting $\Delta K(t)$.

Fig. 6b shows the Euler recurrent neural network cell design to estimate stress intensity range as a function of cyclic stresses and current damage level, while also performing fatigue crack growth integration. At the time t, fatigue damage increment, Δa_t, is quantified through a physics-informed node in the graph while a multilayer perceptron (MLP) implements the stress intensity range, $\Delta K(t)$. With estimated cycle-by-cycle load history, we can integrate Eq. (7) using

$$\begin{aligned} &a_t = a_{t-1} + \Delta a_t, \qquad \Delta a_t = C\Delta K_t^m \text{ , and} \\ &\Delta K_t = \text{MLP}(\Delta S_t, a_{t-1}; \mathbf{w}, \mathbf{b}), \end{aligned} \tag{9}$$

where $\mathbf{w}$ and $\mathbf{b}$ are the multilayer perceptron hyperparameters. Table 1 details the multilayer perceptron architecture. Optimization



(a) Fatigue crack is estimated by obtaining the load spectrum for different missions and then accumulating the damage over the aircraft useful life.

(b) Physics-informed neural network cell for fatigue crack growth.

Fig. 6. Fatigue crack growth application and physics-informed neural network model.

Table 1
Multilayer perceptron architecture for stress intensity range, ΔK_t.

Layer	#1	#2
#neurons/activation function	5/tanh	1/linear

of hyperparameters was carried over up to 100 epochs using the Adamax optimizer [43,44].

The multilayer perceptron takes ΔS_t and a_{t-1} as inputs and returns ΔK_t, which is never observed. Therefore, the training of the physics-informed neural network has to use a, which is available after inspection of the aircraft. Here, we use the mean squared error as loss function, Λ, accounting for fatigue crack length prediction error at inspection:

$$\Lambda = \frac{1}{n}(\mathbf{a} - \hat{\mathbf{a}})^T(\mathbf{a} - \hat{\mathbf{a}}), \tag{10}$$

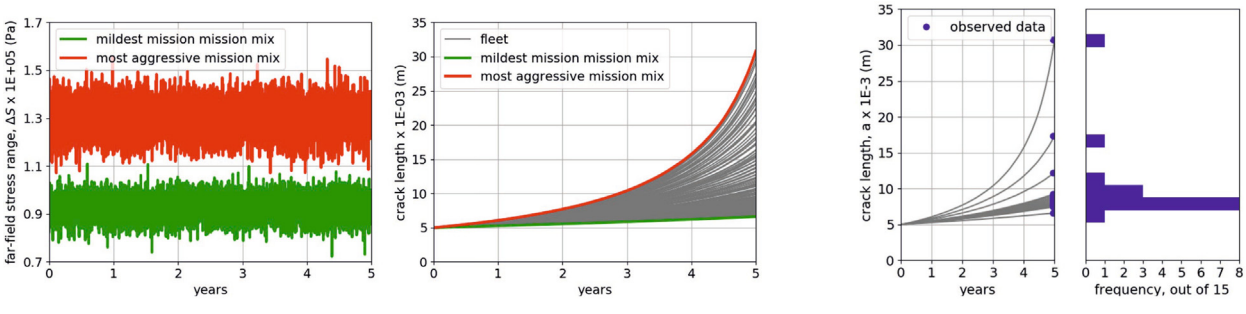
where n is the number of observations, $\mathbf{a}$ are fatigue crack length observations, and $\hat{\mathbf{a}}$ are the predicted fatigue crack length using the physics-informed neural network. The gradients of Λ with regards to $\mathbf{w}$ and $\mathbf{b}$, needed by the stochastic gradient descent optimization, are obtained through automatic differentiation of the graph.

Fig. 7a illustrates all the data used in this study. We consider an aircraft fleet flying 4 flights per day, in a period of 5 years. It shows the complete time histories in terms of both far-field stresses and crack lengths for the entire fleet of 300 aircraft (differences due to different mission mixes). For the sake of training and testing the performance of the physics-informed neural network, we consider that while the history of cyclic loads is known throughout the operation of the fleet, crack length history is not available. Instead, as illustrated in Fig. 7b, observations for crack length are available only for 15 aircraft at the end of the fifth year of operation. This means that we observed 15 time histories of 7300 data points each (total of 109,500 input points) and only 15 output observations.

Fig. 7c illustrates the crack length predictions over time before and after training and how they compare with the actual crack length histories. There is good agreement between predicted and actual crack length histories (also demonstrated through the ratio between the predicted and actual crack length). Initially, the model grossly under-predicts large crack lengths and predictions are within a 35% and 85% of the actual crack length. After the physics-informed neural network is trained, predictions stay within ±15% of the actual crack length, for the most part.

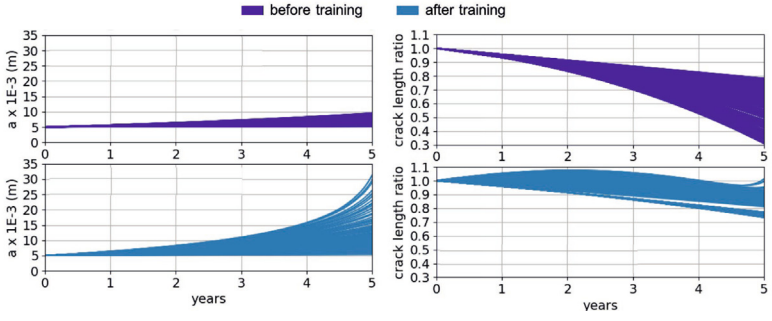
In the second part of this study, we investigate the effect of the distribution of observed crack lengths in the performance of the physics-informed neural network. Fig. 8a highlights four hypothetical cases for selecting aircraft for inspection at the end of the 5th year of operation. Cases #1 and #4 represent the two extreme cases in which only aircraft flying at either the mildest or the most aggressive mission mixes are selected. Case #2 is the case in which the sample of selected 15 aircraft reflects the distribution of the fleet. Case #3 is the case in which the sample of selected 15 aircraft exhibits uniform distribution of crack length. One can argue that cases #1 and #3 are somewhat unlikely to happen in real life. After all, there might be no interest in inspecting aircraft flying only mild missions and it would be a coincidence if inspected crack length distribution is uniform. One can also argue that cases #2 and #4 are somewhat the most likely to happen. Case #2 could be obtained if aircraft are chosen at random from the fleet. Samples similar to case #4 could be obtained by selecting the aircraft that is assigned to the most aggressive missions most of the time (although this would be only an approximation).

Fig. 8b shows the summary of results for this part of the study. Interestingly, except for case #1, the trained physics-informed neural network was able to predict crack length. In fact, there is only minor differences among cases #2, #3, and #4. This indicates that as long as the range of observed crack length covers the plausible crack length range at the fleet level, the resulting model tends to be accurate, and the distribution of observed crack length has minor effects on the resulting network.



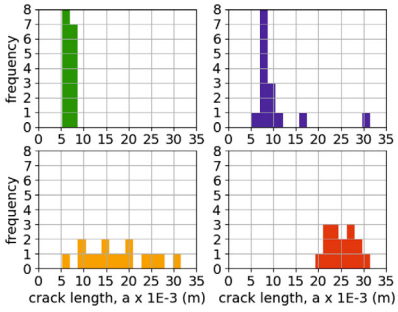
(a) Snapshot of fleet-wide data (300 aircraft).

(b) 15 inspected aircraft.

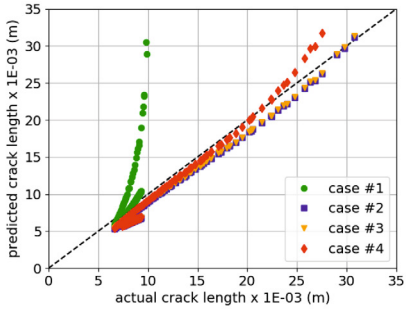


(c) Predicted fatigue crack length over time (300 aircraft).

Fig. 7. Fleet wide data, observed data, and prediction results out of the physics-informed neural network. Crack length ratio is the ratio between predicted and observed crack length.



(a) Observations (15 aircraft) at the 5th year.



(b) Predictions versus actual crack length at the 5th year for entire fleet (300 aircraft).

Fig. 8. Distribution of observed crack length and prediction of physics-informed neural network in the fatigue crack growth application.

4.2. Corrosion-fatigue in aircraft wing

Similarly to the previous case study, we consider a control point on a aircraft wing panel that is inspected at regular intervals through non-destructive evaluation approaches. However, here we assume that upon inspection of first few aircraft, it was found that the failure mechanism is more complex than anticipated. While purely mechanical fatigue was assumed to dominate fatigue crack growth at this control point, inspection revealed that the fleet was subjected to corrosion-fatigue instead.

As inspection data is gathered, the predictive models started been built using a modified Paris law (known as the Walker model) to account for the effect of mean stress [40]:

$$\frac{da}{dt} = \frac{C_0}{(1-R(t))^{m(1-\gamma)}}(\Delta K(t))^m, \tag{11}$$

where $R = S_{min}/S_{max}$ is the stress ratio between the minimum and maximum stress levels and C_0 and m depend on the environmental conditions and γ depends on the material and loading conditions.

Given that operating conditions are available throughout time, integrating Eq. (11) should be straightforward. However, in this application, we consider the case in which there are multiple mechanisms contributing to crack growth. First, there is fatigue induced by mechanical loading, which contributes throughout the useful life of the aircraft. Second, there is corrosion-fatigue, which only contributes to crack growth during take-off and landing (as the aircraft is exposed to potentially corrosive environment during these flight phases).

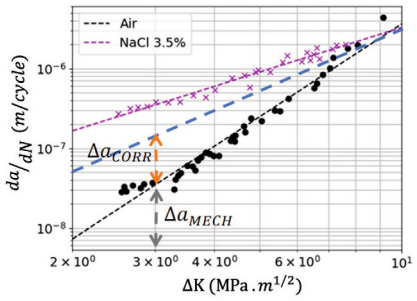
Corrosion-fatigue is a very complex phenomenon involving pit nucleation, pit growth, fatigue crack nucleation, short crack growth, transition from short crack to long crack, and long crack growth [45,46]. Usually, coupon tests are performed to determine how a particular corrosive agent accelerates the damage accumulation. For example, Menan and Henaft [47] showed that the concentration of sodium chloride (NaCl) increases the damage accumulation rate. As illustrated in Fig. 9a, the concentration of NaCl moves the da/dN curve upwards.

In this study, we assume that the only known failure mechanism is fatigue. Therefore, damage would be estimated integrating Eq. (11) using C_0 and m for a non-corrosive environment. However, the fleet of aircraft (flying different route structures) is exposed to corrosive environment at roughly 5% of the cycles. We use an index c to express the NaCl concentration, and therefore, capture the airport to airport variation. Then, we assume that upon investigation of failure cases, we learn that the failure mechanism is corrosion-fatigue. However, we assume we do not know how to model corrosion-fatigue mathematically.

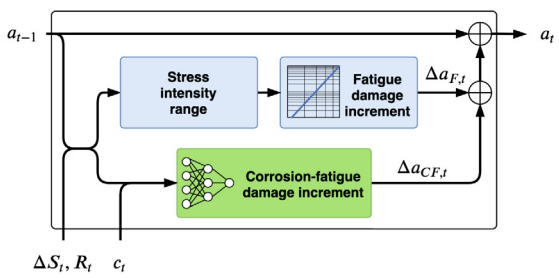
This way, we use a physics-informed neural network to learn the missing physics due to our lack of knowledge about the contribution of corrosion-fatigue towards damage accumulation. Fig. 9b shows the recurrent neural network cell used in this paper. At the time t, fatigue damage increment, Δa_t, is quantified through a physics-informed node in the graph while a multilayer perceptron (MLP) implements the corrosion-fatigue damage accumulation increment. With estimated cycle-by-cycle load history, we can integrate Eq. (11) using

$$\begin{aligned} a_t &= a_{t-1} + \Delta a_t, & \Delta a_t &= \Delta a_{F,t} + \Delta a_{CF,t}, \\ \Delta a_{F,t} &= \frac{C_0}{(1-R_t)^{m(1-\gamma)}}(\Delta K_t)^m, \text{ and} & \Delta a_{CF,t} &= \text{MLP}(\Delta S_t, R_t, c_t, a_{t-1}; \mathbf{w}, \mathbf{b}), \end{aligned} \tag{12}$$

where ΔK_t is the stress intensity range given by Eq. (8) and $\mathbf{w}$ and $\mathbf{b}$ are the multilayer perceptron hyperparameters. Table 2 details the multilayer perceptron architecture. Optimization of hyperparameters was carried over up to 25 epochs using the RMSProp optimizer [44]



(a) Corrosion effect on cyclic damage (data from [47]).



(b) Physics-informed neural network cell.

Fig. 9. Coupon data and physics-informed neural network model for the corrosion fatigue crack growth study.

Table 2
Multilayer perceptron architecture for corrosion fatigue damage increment, Δa_{CF}.

Layer	#1	#2	#3	#4
#neurons/activation function	20/elu	10/sigmoid	5/sigmoid	1/elu

The activation used were the sigmoid and exponential linear unit (elu) functions:

$$\text{sigmoid}(z) = \frac{1}{1+e^{-z}}, \quad \text{elu}(z) = \begin{cases} z & \text{when } z > 0, \text{ and} \\ e^z - 1 & \text{otherwise.} \end{cases} \tag{13}$$

While other architectures and activations can be potentially used here, such optimization study is outside the scope of this paper (one interesting option is to use neural architecture search [48–50] for the design of the data-driven nodes).

The multilayer perceptron takes $\Delta S_t, R_t, c_t, a_{t-1}$ as inputs and returns Δa_{CF}, which is never observed. Therefore, the training of the physics-informed neural network has to use a, which is available after inspection of the aircraft. Here, we use the mean absolute percentage error [51] as loss function, Λ, accounting for the prediction error at inspection:

$$\Lambda = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{a}_i - a_i}{a_i}\right|, \tag{14}$$

where n is the number of observations, a_i are fatigue crack length observations, and $\hat{a}_i$ are the predicted fatigue crack length using the physics-informed neural network.

Similarly to the fatigue crack growth study of Section 4.1, we considered a fleet of 150 aircraft (flying different route structures). For the sake of training and testing, the history of cyclic loads and corrosivity index is known throughout the operation of the fleet. Crack length observations are available for 15 aircraft for training the physics-informed neural network. Fig. 10a shows the predicted crack length at the end of the 5th year of operation (after training). As expected, the predictions out of purely mechanical fatigue grossly underestimates the actual crack length (resulting from corrosion-fatigue damage). On the other hand, the physics-informed neural network is able to adjust the model and reduce significantly the gap between predicted and actual crack length. Although the model is slightly conservative, it successfully ranks the fleet. With that, the predictions can be used to prioritize which aircraft should be inspected next (the ones with the highest predicted crack lengths).

We then studied the ability of the model to forecast corrosion-fatigue damage. We achieve that by running the simulations up to the end of the 6th year of operation. Fig. 10b presents the comparison between predicted and actual crack lengths by that time. Similarly to what happened before, the model is expected to be slightly conservative (predicted values are expected to be higher than the actual ones). Nevertheless, we could use these predictions to assess how many airplanes in the fleet would be above an arbitrary repair/replacement threshold (which is highly dependent on the application). For a threshold of 20 (mm), Fig. 10b shows that there are 22 true positives (only predictions up to 40 mm were illustrated), and 123 true negatives, no false negatives, and only 5 false positives. In practical terms, this means that the model correctly flagged all airplanes with actual crack length above the threshold. Moreover, only 5 airplanes would be flagged for repair/replacement, even though they were not above the threshold (at least 2 of those would be very near the threshold). In safety critical applications, having no false negatives is extremely important since an aircraft being wrongly cleared to flight can have serious implications. Minimizing the false positives is important not only for the credibility of the model, but also to reduce unnecessary inspection costs.

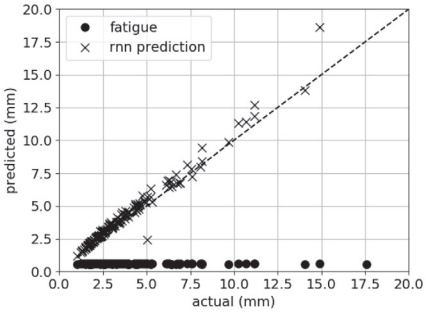
4.3. Wind turbine bearing fatigue

The complexity of design, manufacturing, and operation makes main bearing diagnosis and prognosis a daunting task [52–57]. Unfortunately, a large portion of the operating fleet lacks condition monitoring systems providing anomaly detection through vibration analysis. With that limitation, predictive models for bearing fatigue have to be built with data coming out of supervisory control and data acquisition (SCADA) systems (time series of operation and usage variables such as bearing temperatures, wind speed, yaw angle, and controller flags, etc.) and inspection and services data (scheduled visual inspection, maintenance records, etc.).

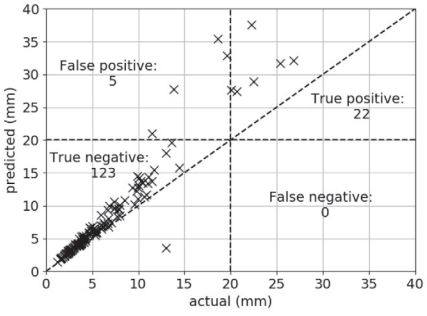
We will use bearing fatigue as baseline model for damage accumulation. For spherical roller bearings operating at different load levels and rotational speeds, fatigue damage, a_{BRG}, is governed by [58]:

$$\begin{aligned} &\frac{da_{BRG}}{dt} = \frac{1}{c_1 c_2(t)}\left(\frac{P(t)}{C}\right)^{\frac{10}{3}}, && P(t) = f_1(V_W(t)), \\ &c_2(t) = f_2(P(t), \eta_c(t), \nu(t)), && \\ &\eta_c(t) = f_3(\nu(t), a_{GRS}(t)), \text{and} && \nu(t) = f_4(T_{BRG}(t), a_{GRS}(t)), \end{aligned} \tag{15}$$

where c_1 is a reliability level factor (see Table 3); c_2 is an adjustment factor; P is the equivalent dynamic bearing load; C is the design load rating; η_c is the grease contamination factor; ν is the viscosity; V_W is the wind speed; T_{BRG} is the bearing temperature; a_{GRS} is an indicator of grease degradation; and $f_{1\ldots4}(.)$ are functions defining the models for different components of the bearing damage.



(a) Fleet prediction at the end of 5th year.



(b) Fleet forecast at the end of the 6th year (only predictions below 40 mm are illustrated).

Fig. 10. Recurrent neural network predictions considering the multilayer perceptron architecture presented in Table 2.

Table 3
c_1 bearing fatigue life adjustment factor [58].

Reliability level (%)	90	95	96	97	98	99
c_1	1.00	0.62	0.53	0.44	0.33	0.21

The model described in Eq. (15) is well understood and widely used in bearing design. Unfortunately, uncertainty in the grease degradation mechanism often hinders the model to be used in fleet management. One can easily recognize grease condition degrades over time and this increases the bearing damage accumulation rate. Therefore, it is important to quantify grease degradation over time.

In this study, we chose a 1.5 MW wind turbine with 80 meters hub height, equipped with a main bearing in the three-point mounting configuration. Design specifications about the wind turbine and main bearing can be found in Refs. [58–60]. P and c_2 vary over time due to wind speed and bearing temperature, but also due to grease condition, contributing to a non-linear bearing damage accumulation. Fig. 11a depicts how bearing load varies with wind speed (i.e., f_1 in Eq. (15)). Fig. 11b illustrates the input-output relationship for η_c, v, and c_2 (i.e., $f_{2\ldots4}$ in Eq. (15)).

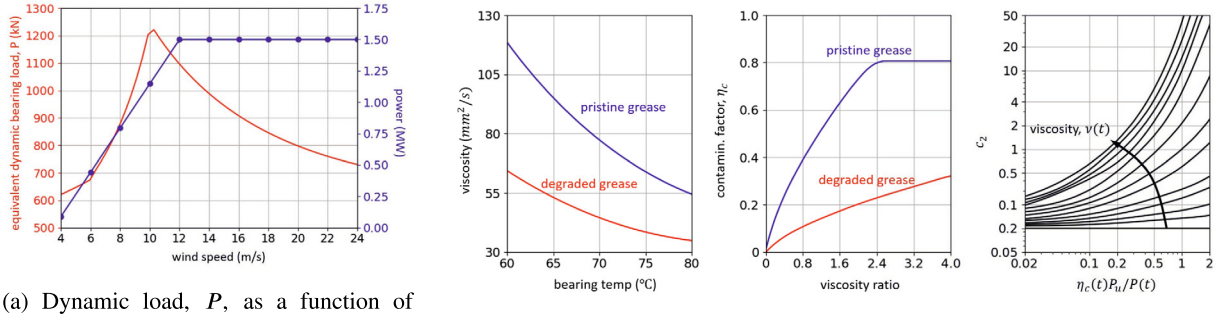
Wind turbine operational data is obtained using a database made available by the U.S. National Renewable Energy laboratory [61], which includes environmental data between 2007 and 2013 for different locations throughout the United States. With that, we have built a set of 14 turbines, for which we observed wind speed, bearing temperature, as well as grease damage data. Fig. 12a illustrates the wind speed and bearing temperature recorded every 10 min over 7 years for one turbine. We use 10 of these turbines for training of our physics-informed neural network model. Fig. 12b summarizes the collected observations of grease damage for these 10 turbines. Grease damage is observed monthly over a six-month period and does not evolve at the same rate across the turbines in the training set due to differences wind speed and main bearing temperature.

Given that operating conditions are available throughout time, the integration of Eq. (15) should be straightforward, except for the fact that quantifying the influence of grease in bearing fatigue is known to be difficult from a purely physics-based perspective [62,63]. Fig. 12c shows the recurrent neural network cell used in this paper. At the time t, bearing damage increment, $\Delta a_{BRG,t}$, is quantified through physics-informed nodes in the graph while a multilayer perceptron (MLP) quantifies the grease damage increment, $\Delta a_{GRS,t}$:

$$\Delta a_{BRG,t} = \frac{1}{c_1 c_{2,t}} \left(\frac{P_t}{C}\right)^{\frac{10}{3}} \quad \text{and} \quad \Delta a_{GRS,t} = \text{MLP}(V_{W,t}, T_{BRG,t}, a_{GRS,t-1}; \mathbf{w}, \mathbf{b}), \tag{16}$$

where $\mathbf{w}$ and $\mathbf{b}$ are the multilayer perceptron hyperparameters. Table 4 details the multilayer perceptron architecture. Optimization of hyperparameters was carried over up to 50 epochs using the RMSprop optimizer.

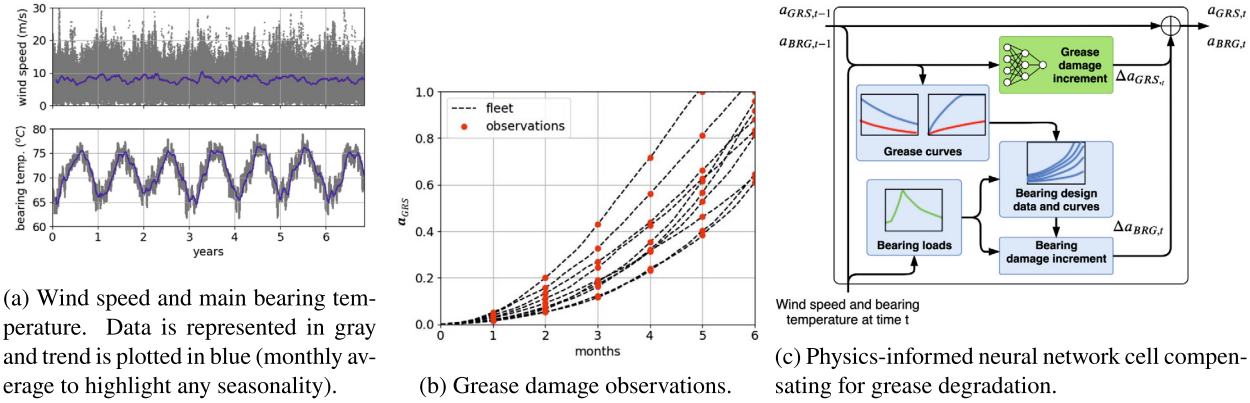
The multilayer perceptron takes $V_{W,t}$, $T_{BRG,t}$, and $a_{GRS,t-1}$ as inputs and returns $\Delta a_{GRS,t}$, which is never observed. Therefore, the training of the physics-informed neural network uses a_{GRS}, which is available through grease inspection. Here, we use the mean squared



(a) Dynamic load, P, as a function of wind speed (obtained through high-fidelity multi-body dynamics analysis [60]).

(b) c_2 adjustment factor as a function of contamination η_c, load P, fatigue limit P_u, and viscosity ν [58].

Fig. 11. Non-linear time-dependent components of the bearing damage model [58,60].



(a) Wind speed and main bearing temperature. Data is represented in gray and trend is plotted in blue (monthly average to highlight any seasonality).

(b) Grease damage observations.

(c) Physics-informed neural network cell compensating for grease degradation.

Fig. 12. Data and physics-informed neural network model used in the wind turbine bearing fatigue application. $a_{GRS,t}$ and $a_{BRG,t}$ are the time-dependent grease and bearing damage, respectively. Blue and green boxes are physics-informed and data-driven nodes, respectively.

Table 4
Multilayer perceptron architecture for grease degradation increment, Δa_{GRS}.

Layer	#1	#2	#3	#4	#5
#neurons/activation function	40/ sigmoid	20/ elu	10/ elu	5/ elu	1/ sigmoid

error as loss function, Λ, accounting for grease damage prediction error at inspection:

$$\Lambda = \frac{1}{n}(\mathbf{a}_{GRS} - \hat{\mathbf{a}}_{GRS})^T(\mathbf{a}_{GRS} - \hat{\mathbf{a}}_{GRS}), \quad (17)$$

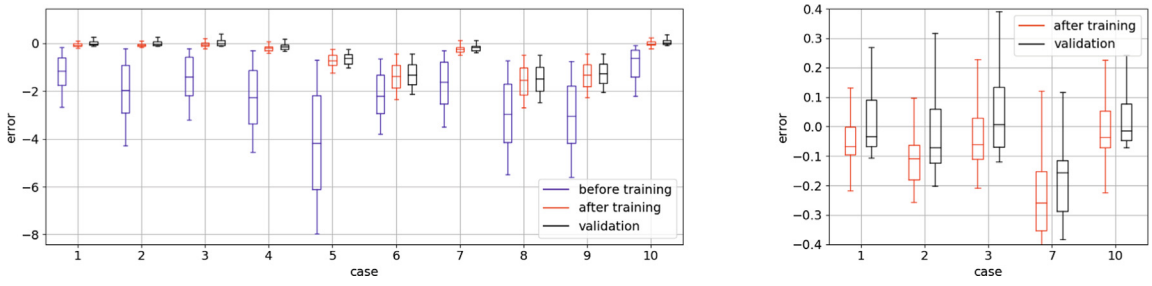
where n is the number of observations, $\mathbf{a}_{GRS}$ are grease damage observations, and $\hat{\mathbf{a}}_{GRS}$ are the predicted grease damage using the physics-informed neural network.

In our study, we repeat the training of the physics-informed neural network 10 times to reduce the influence of the hyperparameter initialization. Fig. 13a summarizes the grease damage prediction errors at both the 10 training and 4 test turbines. The results are very good for 4 out of 10 cases (#1, #2, #3, and #10), fairly well for another 3 cases (#4, #5, and #7), and poor for the remaining 3 cases (#6, #8, and #9). Fig. 13b highlights the prediction errors for the top 5 best cases of parameter initializations (i.e., cases #1, #2, #3, #7, and #10). Overall, hyperparameter initialization can impact the final accuracy of the model. From the perspective of optimizing the maintenance of the turbines, a slight degree of conservatism might be tolerable.

We then used the physics-informed neural network model to estimate grease damage accumulation and main bearing fatigue damage. Fig. 14 illustrates the results of three best and one mediocre training case from previous simulations (cases #1, #3, #10, and #7, respectively). Fig. 14a shows the prediction results of the recurrent neural network predictions and actual grease damage over time. Since the bearing is fully regreased every 6 months, the grease damage is reset back down to zero periodically. The conservatism in a_{GRS} estimation influences bearing fatigue damage accumulation. Fig. 14b illustrates the actual and predicted main bearing fatigue damage accumulation as well as two additional curves bounding bearing fatigue estimation. The solid-red and solid-green lines are obtained when fully degraded and non-damaged greases are used throughout the predictions, respectively. It is intuitive that conservative grease damage predictions result in conservative bearing fatigue life predictions. Nevertheless, the slightly conservative physics-informed neural network models (such as case #1, #3, and #10) predict bearing fatigue failure a few months earlier than when it actually happens (out of roughly 16 years of total life).

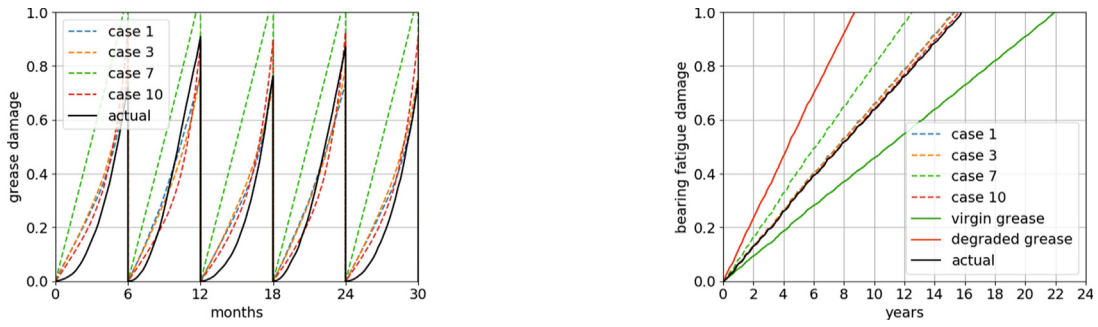
4.4. Replication of results

The implementations discussed in this paper are all done in TensorFlow (version 2.0.0-beta1) using the Python application programming interface. In order to replicate our results, the interested reader can download codes and data. First, install the PINN package (base package for physics-informed neural networks used in this work) available at [64]. For each specific application, the interested reader is referred to [64–66]. In the applications described in this paper, we determined the choise of multilayer perceptron architecture, loss function, and optimization setup (e.g., optimization algorithm, number of epochs, learning rate, etc.) using trial runs on a case-by-case basis. This strategy can be improved in the future with the use of technologies such as neural architecture search [48–50].



(a) All 10 cases. (b) Best 5 cases.

Fig. 13. Box plot for grease damage prediction errors, computed as $a_{GRS} - \hat{a}_{GRS}$, where a_{GRS} and $\hat{a}_{GRS}$ are the observed and predicted grease damage, respectively. Cases come from different initialization of multilayer perceptron.



(a) Grease damage, d^{GRS}. (b) Bearing damage, d^{BRG}.

Fig. 14. Predicted and actual damage over time.

5. Summary and conclusions

In this work, we proposed a framework for reducing the gap between predictions and observations of systems described by ordinary differential equations. Throughout the paper, we use the terms missing physics to refer to the model-form uncertainty responsible for the discrepancies between predictions and observations. Our framework is based on the direct implementation of physics-informed models within recurrent neural networks. In order to achieve that, we first prescribe the design of recurrent neural network cells such that specific numerical integration methods (e.g., Euler, Riemann, Runge-Kutta, etc.) are implemented. Then, we model the recurrent neural network as a directed graph, where nodes represent specific kernels of the physics-informed model. With the graph built for the physics-informed model, we can add data-driven nodes (implemented as multilayer perceptrons, for example) to correct the outputs of specific nodes in the graph, reducing model discrepancy.

In order to evaluate the performance of the proposed framework, we presented three case studies with different manifestations of model-form uncertainty. Throughout the three case studies, we used a Euler recurrent neural network for numerical integration. The first case study consisted of fatigue crack growth estimation in which uncertainty in stress intensity range can lead to disagreement between predicted and observed crack length over time. The recurrent neural network cell implemented fatigue damage increment through a series arrangement of two nodes. The first node implemented the stress intensity range through a multilayer perceptron and the second node implemented Paris law. The second case study consisted of corrosion-fatigue crack growth estimation in which uncertainty was driven by lack of knowledge of physics of failure (as model initially neglected corrosion-fatigue contribution). The physics-informed nodes implemented mechanical fatigue damage increment through Walker's model. The data-driven node compensated for the underestimation of damage in corrosive cycles. The last case study consisted of main bearing fatigue estimation in which uncertainty was driven by lack of knowledge about grease degradation. The physics-informed nodes implemented bearing fatigue damage increment. The data-driven node modeled grease degradation on a cycle-by-cycle basis. In this case study, both bearing fatigue and grease degradation are simultaneously integrated over time (as grease degradation increases bearing fatigue damage accumulation rates). The model architecture alternatives (number of layers, number of neurons, activations functions, and choice of optimizers) of the data-driven nodes are outside the scope of this paper. Subject to application complexity, one could pursue the optimization of the data-driven architecture alternatives with technologies such as neural architecture search [48–50].

With the help of these case studies, we learned that:

1. *Our approach is flexible when accounting for model-form uncertainty:* The first two case studies showed missing physics estimation where data-driven nodes were put in either series or parallel arrangements with physics-informed nodes. The third case study showed missing physics estimation with two states (bearing fatigue and grease degradation) that interact with each other throughout time.
2. *Our approach can handle highly unbalanced datasets:* In all case studies, the inputs were observed the input variables for few assets (aircraft or wind turbines depending on the case) throughout long periods of time (thousands of data points). However, the outputs were observed only at inspection points (one data point per asset). Despite of that, the fact that physics-informed nodes were present in the recurrent neural network cell seemed to have penalized unfeasible solutions and guided the hyperparameter optimization.
3. *Resulting hybrid models can be used for diagnosis and prognosis:* we used the obtained models to predict different damage metrics both at current time (diagnosis) but also in forecast (prognosis). This way, they can be used in fleet management tasks such as prioritization of inspection and maintenance planning.

Despite the many desirable features, our proposed methodology has the following deficiencies:

1. *Computational cost highly dependent on physics-informed nodes:* in general terms, we do not see problems if computational cost of different nodes in the graph are comparable to the linear algebra typically found in neural networks. However, computational cost can become a problem as complexity of physics-informed nodes increases. Some remedy can be found in reduced-order modeling approaches.
2. *Backpropagation relies on readily available gradients:* although most modern computational libraries and frameworks for deep learning have inherent automatic differentiation capabilities, this has to be kept in mind specially with customized implementations.
3. *Model predictions do not necessarily pass through the points at the training (nor at the test/validation) points:* depending on the application, this can lead to either undesired conservatism or under prediction.

While the obtained results enabled us to reduce discrepancy between predictions and observed data, we see that one can extend our framework to account for other sources of uncertainty. For example, one can study the uncertainty in the loads model to include limitations in assumptions of finite element models used in estimation of far-field loads. Additionally, one can also study how to account for scatter in material properties, including limitations in coupon data. Moreover, capabilities of proposed framework to solve partial differential equations also can be explored.

CRediT authorship contribution statement

Felipe A.C. Viana: Conceptualization, Methodology, Validation, Software, Formal analysis, Investigation, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition. **Renato G. Nascimento:** Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Arinan Dourado:** Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Yigit A. Yucesan:** Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization.

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.

References

[1] Möller B, Beer M. Engineering computation under uncertainty – capabilities of non-traditional models. Comput Struct 2008;86:1024–41.
[2] Riley ME, Grandhi RV. Quantification of model-form and predictive uncertainty for multi-physics simulation. Comput Struct 2011;89:1206–13.

[3] Kennedy MC, O'Hagan A. Bayesian calibration of computer models. J Roy Stat Soc Ser B (Stat Methodol) 2001;63:425–64.
[4] McFarland J, Mahadevan S, Romero V, Swiler L. Calibration and uncertainty analysis for computer simulations with multivariate output. AIAA J 2008;46:1253–65.
[5] Peherstorfer B, Willcox K, Gunzburger M. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Rev 2018;60:550–91.
[6] Zhang D, Lu L, Guo L, Karniadakis GE. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. J Comput Phys 2019;397:108850.
[7] Raissi M, Perdikaris P, Karniadakis G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[8] Raissi M. Deep hidden physics models: deep learning of nonlinear partial differential equations. J Mach Learn Res 2018;19:1–24.
[9] Wu J-L, Xiao H, Paterson E. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. Phys Rev Fluids 2018;3:074602.
[10] Hesthaven JS, Ubbiali S. Non-intrusive reduced order modeling of nonlinear problems using neural networks. J Comput Phys 2018;363:55–78.
[11] Swischuk R, Mainini L, Peherstorfer B, Willcox K. Projection-based model reduction: Formulations for physics-based machine learning. Comput Fluids 2019;179:704–17.
[12] Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V, Guo H, Hamdia K, et al. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Comput Methods Appl Mech Eng 2020;362:112790.
[13] Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. Theoret Appl Fract Mech 2020;106.
[14] Jardine AKS, Lin D, Banjevic D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mech Syst Sig Process 2006;20:1483–510.
[15] Barber D. Bayesian Reasoning and Machine Learning. Cambridge University Press; 2012.
[16] Murphy KP. Machine learning: a probabilistic perspective. MIT Press; 2012.
[17] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016. http://www.deeplearningbook.org.
[18] Pearlmutter BA. Learning state space trajectories in recurrent neural networks. Neural Comput 1989;1:263–9.
[19] Aussem A. Dynamical recurrent neural networks towards prediction and modeling of dynamical systems. Neurocomputing 1999;28:207–32.
[20] Chauhan S, Vig L. Anomaly detection in ECG time signals via deep long short-term memory networks, in. In: IEEE international conference on data science and advanced analytics (DSAA). p. 1–7. https://doi.org/10.1109/DSAA.2015.7344872.
[21] Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks. In: IEEE international conference on acoustics, speech and signal processing. p. 6645–9. https://doi.org/10.1109/ICASSP.2013.6638947.
[22] Yuan M, Wu Y, Lin L. Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In: 2016 IEEE international conference on aircraft utility systems (AUS). IEEE; 2016. p. 135–40. https://doi.org/10.1109/AUS.2016.7748035.
[23] Guo L, Li N, Jia F, Lei Y, Lin J. A recurrent neural network based health indicator for remaining useful life prediction of bearings. Neurocomputing 2017;240:98–109.
[24] You G-W, Park S, Oh D. Diagnosis of electric vehicle batteries using recurrent neural networks. IEEE Trans Industr Electron 2017;64:4885–93.
[25] Wu Q, Ding K, Huang B. Approach for fault prognosis using recurrent neural network. J Intell Manuf 2018:1–13.
[26] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9:1735–80.
[27] Zhang R, Chen Z, Chen S, Zheng J, Büyüköztürk O, Sun H. Deep long short-term memory networks for nonlinear structural seismic response prediction. Comput Struct 2019;220:55–68.
[28] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078; 2014.
[29] Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical Recipes: The Art of Scientific Computing, vol. 3. Cambridge: Cambridge University Press Cambridge; 2007.
[30] Amsallem D, Farhat C. Interpolation method for adapting reduced-order models and application to aeroelasticity. AIAA J 2008;46:1803–13.
[31] Benner P, Cohen A, Ohlberger M, Willcox K. Model reduction and approximation: theory and algorithms,, vol. 15. SIAM; 2017.
[32] Baydin AG, Pearlmutter BA, Radul AA, Siskind JM. Automatic differentiation in machine learning: a survey. J Mach Learn Res 2018;18:1–43.
[33] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI). p. 265–83.
[34] Ketkar N. Introduction to PyTorch. Berkeley, CA: Apress; 2017. p. 195–208. https://doi.org/10.1007/978-1-4842-2766-4_12.
[35] Chollet F, et al. Keras; 2015. https://keras.io.
[36] Aldrin JC, Knopp JS, Lindgren EA, Jata KV. Model-assisted probability of detection evaluation for eddy current inspection of fastener sites. In: AIP Conference Proceedings, vol. 1096. AIP; 2009. p. 1784–91. doi:10.1063/1.3114175. URL https://aip.scitation.org/doi/abs/10.1063/1.3114175.
[37] Drinkwater BW, Wilcox PD. Ultrasonic arrays for non-destructive evaluation: A review. NDT & E Int 2006;39:525–41.
[38] Hoye MV. Fluorescent penetrant crack detection, Patent: US4621193A; 1986.
[39] Paris P, Erdogan F. A critical analysis of crack propagation laws. J Basic Eng 1963;85:528–33.
[40] Dowling NE. Mechanical Behavior of Materials: Engineering Methods for Deformation, Fracture, and Fatigue. Pearson; 2012.
[41] Boresi AP, Schmidt RJ. Advanced Mechanics of Materials. 6th ed. John Wiley and Sons; 2003.
[42] Collins JA. Failure of Materials in Mechanical Design: Analysis, Prediction, Prevention. John Wiley & Sons; 1993.
[43] Kingma DP, Ba J. Adam: A method for stochastic optimization. arxiv 2014, arXiv preprint arXiv:1412.6980; 2014.
[44] Zou F, Shen L, Jie Z, Zhang W, Liu W. A sufficient condition for convergences of Adam and RMSprop, in. In: Proceedings of the IEEE conference on computer vision and pattern recognition. p. 11127–35.
[45] Goswami TK, Hoeppner DW. Pitting corrosion fatigue of structural materials. In: Structural integrity in aging aircraft. New York: ASME; 1995. p. 47.
[46] Shi P, Mahadevan S. Damage tolerance approach for probabilistic pitting corrosion fatigue life prediction. Eng Fract Mech 2001;68:1493–507.
[47] Menan F, Henaft G. Synergistic action of fatigue and corrosion during crack growth in the 2024 aluminum alloy. Procedia Eng 2010;2:1441–50.
[48] Kandasamy K, Neiswanger W, Schneider J, Poczos B, Xing EP. Neural architecture search with Bayesian optimisation and optimal transport. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in neural information processing systems 31. Curran Associates, Inc.; 2018. p. 2016–25.
[49] Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li L-J, et al. Progressive neural architecture search. In: The European Conference on Computer Vision (ECCV), Computer Vision Foundation, Munich, Germany; 2018.
[50] Elsken T, Metzen JH, Hutter F. Neural architecture search: a survey. J Mach Learn Res 2019;20:1–21.
[51] Goodwin P, Lawton R. On the asymmetry of the symmetric MAPE. Int J Forecast 1999;15:405–8.
[52] Hornemann M, Crowther A. Establishing failure modes for bearings in wind turbines, Online (retrieved 16 Oct 2020); 2013. https://www.windpowerengineering.com/establishing-failure-modes-for-bearings-in-wind-turbines/.
[53] Siegel D, Zhao W, Lapira E, AbuAli M, Lee J. A comparative study on vibration-based condition monitoring algorithms for wind turbine drive trains. Wind Energy 2014;17:695–714.
[54] Tautz-Weinert J, Watson SJ. Using scada data for wind turbine condition monitoring–a review. IET Renew Power Gener 2016;11:382–94.
[55] Maheswari RU, Umamaheswari R. Trends in non-stationary signal processing techniques applied to vibration analysis of wind turbine drive train–a contemporary survey. Mech Syst Signal Process 2017;85:296–311.
[56] Yucesan YA, Viana FAC. Onshore wind turbine main bearing reliability and its implications in fleet management. In: AIAA Scitech 2019 Forum. Orlando, USA: AIAA; 2019. https://doi.org/10.2514/6.2019-1225. AIAA-2019-1225.
[57] Hart E, Clarke B, Nicholas G, Amiri AK, Stirling J, Carroll J, et al. A review of wind turbine main bearings: design, operation, modelling, damage mechanisms and fault detection. Wind Energy Sci 2020;5:105–24.
[58] SKF-contributors. SKF spherical roller bearings catalogue, Online (retrieved 5 June 2018); 2007. http://www.skf.com/binary/30-148465/6100_EN.pdf.
[59] GE-contributors. GE Energy 1.5 MW Wind Turbine Brochure, Online (retrieved 23 May 2018); 2009. https://geosci.uchicago.edu/moyer/GEOS24705/Readings/GEA14954C15-MW-Broch.pdf.
[60] Sethuraman L, Guo Y, Sheng S. Main bearing dynamics in three-point suspension drivetrains for wind turbines. In: American wind energy association conference & exhibition, AWEA, Orlando, USA; 2015.
[61] Draxl C, Clifton A, Hodge B-M, McCaa J. The wind integration national dataset (wind) toolkit. Appl Energy 2015;151:355–66.
[62] Iyer NS, Qiu H, Yan W, Loparo KA. Early detection of lubrication anomalies in oil-lubricated bearings. In: ASME Turbo Expo 2007: Power for Land, Sea, and Air. Montreal, Canada: American Society of Mechanical Engineers Digital Collection; 2007. p. 785–94. https://doi.org/10.1115/GT2007-27950.
[63] Zhu J, Yoon JM, He D, Qu Y, Bechhoefer E. Lubrication oil condition monitoring and remaining useful life prediction with particle filtering. Int J Prognostics Health Manage 2013;4:124–38.
[64] Viana FAC, Nascimento RG, Yucesan Y, Dourado A. Physics-informed neural networks package; 2019. doi: 10.5281/zenodo.3356877. https://github.com/PML-UCF/pinn.
[65] Dourado A, Viana FAC. Python scripts for physics-informed neural networks for corrosion-fatigue prognosis, v0.0.1; 2019. doi: 10.5281/zenodo.3355729. https://github.com/PML-UCF/pinn_corrosion_fatigue.
[66] Yucesan YA, Viana FAC. Python scripts for wind turbine main bearing fatigue life estimation with physics-informed neural networks; 2019. doi: 10.5281/zenodo.3355725. https://github.com/PML-UCF/pinn_wind_bearing.