Physics-informed neural networks for missing physics estimation in cumulative damage models: a case study in corrosion fatigue

Arinan Dourado
Research Assistant
Felipe A. C. Viana∗
Assistant Professor
Department of Mechanical and Aerospace Engineering
University of Central Florida
Orlando, Florida, 32816
Email: arinandourado@knights.ucf.edu, viana@ucf.edu

ABSTRACT
We present a physics-informed neural network modeling approach for missing physics estimation in cumulative damage models. This hybrid approach is designed to merge physics-informed and data-driven layers within deep neural networks. The result is a cumulative damage model in which physics-informed layers are used to model relatively well understood phenomena and data-driven layers account for hard-to-model physics. A numerical experiment is used to present the main features of the proposed framework. The test problem consists of predicting corrosion-fatigue of an Al 2024-T3 alloy used on panels of aircraft wings. Besides cyclic loading, panels are also subjected to saline corrosion. In this case, physics-informed layers implement the well known Walker model for crack propagation, while data-driven layers are trained to compensate the bias in damage accumulation due to the corrosion effects. The physics-informed neural network is trained using full observation of inputs (far-field loads, stress ratio and a corrosivity index defined per airport) and very limited observation of outputs (crack length at inspection for only a small portion of the fleet). Results show that the physics-informed neural network is able to learn how to compensate the missing physics of corrosion in the original fatigue model. Predictions from the hybrid model can be used in fleet management, for example, to prioritize inspection across the fleet or forecast ahead of time the number of planes with damage above a threshold.

Nomenclature

\( a_t \) crack length at time \( t \).
\( C, m \) Paris law coefficients.
\( C^{IDX} \) environment corrosion index.
\( F \) stress intensity range geometry factor.
\( R_t \) stress ratio at time \( t \).
\( \gamma \) Walker model coefficient.
\( \Delta a_t \) crack length increment at time \( t \).
\( \Delta a_{MECH} \) crack length increment due to mechanical loads.
\( \Delta a_{CORR} \) crack length increment due to mechanical loads under corrosive environment.
\( \Delta K_t \) stress intensity range at time \( t \).
\( \Delta S_t \) far-field stress at time \( t \).

∗Address all correspondence to this author.
1 Introduction

When building prognostic and health management models, we usually focus on characterizing and forecasting the damage state of a system or its components. This damage information can be used then for monitoring systems in order to prevent machine performance degradation, malfunction, or unexpected failures. This predictive model is the core of the prognostic and health management analysis and can be derived in many different ways [1].

A typical model-based approach involves building a model to describe the physics of failure and predict remaining useful life. Usually, the physics-based model considering nominal and degraded conditions is associated with a probabilistic model to predict the system degradation (see [2–4]). Nonetheless, model-based approaches require accurate, reliable, and in some cases, high-fidelity analytical models of the system. These models can be hard to derive or computationally expensive, which can impose a challenge to a purely model-based approach.

The data-driven approach is a possible alternative for prognostics models when analytical models of the system are not available. In a data-driven framework, regression or pattern recognition models are used to establish correlations between operational inputs and system health. The recent growth in computational power has triggered interest in machine learning techniques for prognosis such as dynamic Bayesian networks [5, 6] and artificial neural networks [7–9], from which recurrent neural networks gained more traction in prognosis and health management applications [10–14] due to its ability to handle time-series forecast. Samanta and Nataraj [11] presented a comparative study of different machine learning techniques for machinery condition monitoring and prognosis. They compared the performance of a recurrent neural network, an adaptive neurofuzzy inference system, and support vector regression for prognostic predictions of a helicopter drivetrain gearbox based on vibration signals. For their case study, authors observed that support vector regression performed the best; however, they also highlighted that the improved prediction capability comes at the expense of higher computational cost. Su et al. [14] proposed a deep recurrent neural network for failure prognosis. The proposed framework extract features from time and frequency domains of vibration signals to assess current degradation state. The authors evaluated the performance of the proposed methodology by means of six benchmark datasets of large machinery bearings. Al-Dulaimi et al. [15] proposed a hybrid deep neural network for remaining useful life estimation. The proposed framework merges noisy convolutional and long-short term memory neural networks to extract and learn the dependencies in the input data. In order to improve robustness, the proposed neural network is trained with noisy data. The authors then tested the resulting model on the commercial modular aero-propulsion system simulation (C-MAPSS) dataset provided by NASA.

Despite its advantages (mainly its generalization capabilities), the requirement of significant data sets of failure data (not always available) and lack of clear physical interpretation of the system degradation process (hindering in some cases the ability to propose mitigation strategies) are major drawbacks of data-driven approaches [16, 17]. Hence, a yet third option for building predictive models gained attention in recent years: hybrid models combining model-based and data-driven approaches. The goal of such approach is to reduce the data requirement of data-driven models by taking advantage of physics-based model predictions. Eker et al. [18] presented a hybrid prognosis model for remaining useful life (RUL) estimation, by integrating short-term predictions of physics-based models with long-term estimates of data-driven model. In their approach, system health state progression are represented by physics-based models expressed in the form of state transition equations and tunned within a particle filter framework. Upon these health state predictions the data-driven model estimates remaining useful life based on a similarity analysis of the weighted sum of previously degraded assets. Authors tested their approach by means of two engineering problems, fatigue crack growth and filter clogging. Chao et al. [19] proposed a hybrid methodology that combines physical performance models with deep learning algorithms for fault detection. The system physical model responses are combined with measured data to enhance the input space of a deep-learning diagnostic model. This extended representation of the input space allows the neural-network to infer components deterioration by means of the tuned physical model parameters degradation. The authors validated the proposed approach considering data sets of a gas turbine during flight conditions under healthy and faulty operative conditions.

In this work, we propose a hybrid approach designed to merge physics-informed and data-driven layers within recurrent neural networks. This extends the physics-informed recurrent neural network model introduced by Nascimento and Viana [20, 21], in which, a recurrent neural network cell was proposed to specifically account for damage integration in cumulative damage models. In such design, the recurrent neural network state represents the damage level at a given time step, and a custom cell is used to implement physics-based, data-driven or hybrid cumulative models to map inputs and previously accumulated damage into a damage increment. In this contribution, we proposed to expand this custom cell into a hybrid model in which the data-driven layers are trained to act as bias estimators and compensate for the missing physics of reduced-order cumulative damage model. We use a numerical experiment to present the main features of the proposed physics-informed neural network for missing physics compensation.

The test problem consists of predicting corrosion-fatigue of an Al 2024-T3 alloy used on panels of aircraft wing (one of the main causes of failure of aluminum alloys components [28]. We propose modeling mechanical (load-driven) fatigue crack growth through physics and incorporate a pure data-driven approach to account for the missing physic of corrosion. The
physics-informed layers implement the Walker’s model for crack propagation, while the data-driven layers are trained to act as a bias estimator compensating for the missing physics (increased corrosion-induced damage accumulation on top of purely mechanical fatigue). The numerical integration of damage is done through recurrent neural networks.

The remaining of this paper is organized as follows. A brief discussion of the proposed damage accumulation model is presented in Section 2. The considered numerical experiment and the considered synthetic fleet data are discussed in Section 3. The main results regarding the proposed physics-informed recurrent neural network for fleet prognosis are presented and discussed in Section 4. Finally, some final remarks are addressed in Section 5.

2 Physics-informed Neural Network for Corrosion-fatigue Damage Accumulation

Applying artificial neural networks to solve differential equations in engineering is a well-developed methodology [29--32]. Nevertheless, the computational power available these days contributed to the popularization of machine learning in engineering applications. The scientific community has been studying and proposing deep learning architectures that leverage mathematical models based in physics and engineering principles [33--37]. Physics laws (in the form of differential equations) help handling the reduced number of data points, and constrain the hyper-parameter space. Raissi [38] approximated the unknown solution of partial differential equations by two deep neural networks. The first network acts as a prior on the unknown solution (enabling it to avoid ill-conditioned and unstable numerical differentiations). 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. Yu et al. [39] developed a physics based learning methodology to simulate aircraft dynamics based on a deep residual recurrent neural network. The authors illustrated their methodology by means of a six degrees-of-freedom Boeing 747-100 aircraft model. The model was trained with simulated data of the aircraft model and then used to predict aircraft response under arbitrary control inputs and disturbances.

In the next two subsections, we present cumulative damage models (and the specialized case of corrosion-fatigue). These models will motivate our approach to physics-informed neural networks. Then, we detail how to compensate for bias in the physics-informed layers of a recurrent neural network.

2.1 Cumulative Damage Models and Corrosion-fatigue

Cumulative damage models [40, 41] describe the irreversible accumulation of damage throughout the useful life of components (or systems) such that the damage at time \( t \), \( a_t \), is the sum of a damage increment \( \Delta a_t \) on top of damage \( a_{t-1} \) at previous time step \( t-1 \)

\[
a_t = a_{t-1} + \Delta a_t \tag{1}
\]

where \( \Delta a_t \) is the damage increment, which is often a function of \( a_{t-1} \) and other inputs \( x_t \) at time \( t \).

The characterization of the damage \( a_t \) and the inputs \( x_t \) is highly problem dependent. The damage \( a_t \) is usually associated with a failure mechanism and \( a_t \) is ideally an observable quantity \(^2\). For example, if fatigue is the failure mechanism, crack length is the observable quantity. The inputs \( x_t \) usually express time-dependent loading and boundary conditions (e.g., pressures, temperatures, torques, mechanical and thermal stresses, etc.) or even operating points (e.g., altitude, thrust, angle of attack, etc.). For that matter, fatigue crack propagation is usually modeled through Paris law [43]

\[
da/dt = C(\Delta K)^m \tag{2}
\]

where \( \Delta K \) is the stress intensity range and \( C \) and \( m \) are material properties determined through coupon data and many engineering materials have constants documented in handbooks such as [44].

The Paris law can be modified to take into account the effect of mean stress, expressed in terms of the stress ratio between the minimum and maximum stress levels \( R = S_{\text{min}}/S_{\text{max}} \). We use the Walker model [45] to incorporate the stress ratio:

\[
da/dt = \frac{C_0}{(1-R)^m(1-\gamma)}(\Delta K)^m \tag{3}
\]

where \( C_0 \) and \( m \) depend on the environmental conditions and \( \gamma \) depends on the material and loading conditions. When cycle-by-cycle information is available, the discrete form of Eq. 3 resembles Eq. 1 and becomes:

\[
a_t = a_{t-1} + \frac{C_0}{(1-R)^m(1-\gamma)}(\Delta K)^m, \tag{4}
\]

\(^2\)Although this is not a requirement, it significantly facilitates the modeling task [2,41,42].
where stress intensity range, $\Delta K$ comes from engineering models that use information about geometry, crack length, and internal loads (stress amplitude). For example, assuming that fatigue damage accumulates under mode I loading condition, the stress intensity range $\Delta K_t$ can be expressed as:

$$
\Delta K_t = F \Delta S_t \sqrt{\pi a_t - 1}
$$

where $F$ is a geometry factor and $\Delta S_t$ is the far-field stress.

As we mentioned before, 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. Modeling corrosion-fatigue has proven to be a daunting task [46, 47] and it is not our objective to propose a new method (nor defend any preferred method) for it. Instead, we will focus on how a deep neural network can be designed and trained to act as a bias estimator, compensating the corrosion effects in crack growth.

In this paper, we assume that the corrosivity of an environment can be associated with an index $C_{IDX}$ (such as the concentration of a particular corrosive agent in the air). Empirical studies, such as the one found in [48], are usually necessary to determine how a particular corrosive agent accelerates the damage accumulation. They showed that corrosivity, that in this case is given by the concentration of sodium chloride (NaCl) can drastically increase the damage accumulation rate (shifting the $da/dN$ vs $\Delta K$ curve up, when compared to the rates in pure air), as illustrated in Figure 1a.

We assume that, at varying NaCl concentrations, the damage increment per cycle $\Delta a$ as a function of $\Delta K$ can be modeled as a baseline contribution, $\Delta a_{MECH}$, plus a shift due to corrosion, $\Delta a_{CORR}$ (see Figure 1b), and Eq. 4 can be re-written as

$$
a_t = a_{t-1} + \Delta a_{MECH} + \Delta a_{CORR}
$$

$$
a_t = a_{t-1} + \frac{C_0}{(1 - R_t)^{m(1-\gamma)}} (\Delta K_t)^m + g(\Delta S_t, R_t, C_{IDX}, a_{t-1})
$$

where $g(.)$ is an unknown function. We discuss how we model $g(.)$ in the next section.

Fig. 1: Corrosion-fatigue crack propagation rate. Paris law coefficients when $R = 0$: for pure air $C = 1.132 \times 10^{-10}$ and $m = 3.859$; for NaCl (sodium chloride) at 3.5 % solution $C = 2.241 \times 10^{-8}$ and $m = 1.853$.

2.2 Physics-informed Neural Networks for Cumulative Damage Modeling

A recurrent neural network [49] repeatedly apply transformations to given states in a sequence, as shown in Eq. 7 and illustrated in Figure 2a.

$$
a_t = f(x_t, a_{t-1})
$$

where $t \in [0, \ldots, T]$ represent the time discretization, $a \in \mathbb{R}^{n_a}$ are the states representing the sequence, $x \in \mathbb{R}^{n_x}$ 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.
As illustrated in Figure 2b, a cell implementing \( f(.) \) can be as simple as a single-layer perceptron. However, it can also assume sophisticated forms such as the long short-term memory [50] and the gated recurrent unit [51]. Besides improving generalization, these architectures improve training by mitigating the vanishing/exploding gradient problem [49].

Recurrent neural networks have been successfully used in many sequence modeling applications [52–56]. Nascimento and Viana [21] proposed the cell illustrated in Figure 2c specifically for the integration of cumulative damage. In such design, the state represents the cumulative damage at time \( t \) and “MODEL” maps the inputs \( x_t \) and previously accumulated damage \( a_{t-1} \) into a damage increment \( \Delta a_t \) (equivalent re-writing Eq. 7 as Eq. 1). In a purely physics-based approach, “MODEL” is the computational implementation of the physics of failure (which, again, is highly application dependent). Nevertheless, in the design cell, nothing prevents “MODEL” to be a data-driven model, such as a multi-layer perceptron, or a hybrid model, where some parts are physics-based and others are data-driven.

![Fig. 2: Recurrent neural network, perceptron as the simplest cell, and cumulative damage cell.](image)

One might be tempted to ask why use the cell shown in Fig. 2c and why use physics-informed “MODELS” within that cell? The answer to both questions is in the niche of applications we target in this paper. Our application of cumulative damage models involves full observation of input conditions (i.e., loads), and partial observation of damage (e.g., crack length). This is the case for example when through detailed flight tracking in conjunction with engineering models the loads can be estimated cycle-by-cycle, and a control point in an aircraft fuselage panel is inspected in regular intervals through visual inspection or nondestructive evaluation approaches (e.g., Eddy current [57], ultrasound [58], dye penetrant inspection [59], etc.). Figure 3a shows the typical data collected for training the cumulative damage model. In fairness, this represents one specific aircraft, and likely, there would be load history and inspection data available for few aircraft. Figure 3b illustrates the typical data collected for prediction using the trained cumulative damage model. As we mentioned, load history is available throughout the useful life and the initial value for the states is either known or assumed. The cumulative damage is then used to estimate the damage growth over time. Given the few observations of damage, we argue that using purely data-driven architectures (such as long short-term memory and the gated recurrent unit) is unlikely to lead to accurate models. Nevertheless, the long short-term memory and the gated recurrent unit architectures can be useful in cases where there is a full observation of the states. This happens when control points are continuously monitored with dedicated sensors (e.g., comparative vacuum monitoring [60], fiber Bragg grating sensors [61], etc.). However, this is not the focus of this paper.

![Fig. 3: Typical use-case of recurrent neural network for cumulative damage model.](image)
As previously discussed, the damage accumulation model presented in Eq. 6 is composed of two main terms. The base damage increment is set at \( \Delta a_{MECH} \) and can be described by Eq. 3. Added to the base damage increment, there is a contribution due to corrosion (\( \Delta a_{CORR} \)) (with an unknown functional form). In this paper, we propose the repeating recurrent neural network cell illustrated in Figure 4 to implement the corrosion-fatigue crack growth model shown in Eq. 6. We implemented the Walker model to account for the base \( \Delta a_{MECH} \). This portion of the model (highlighted in blue) is considered physics-informed. Then, we use a multi-layer perceptron to model the dependence between the previous state value (\( a_{t-1} \)) and cycle-by-cycle inputs such as the far-field stress (\( \Delta S \)), stress ratio (\( R \)) and corrosivity index (\( C^{IDX} \)). The multi-layer perceptron is adjusting the damage increment output by the Walker model so that it compensates for the effect of corrosion. Therefore, it works as a discrepancy-correction term.

![Fig. 4: Proposed recurrent neural network cell for corrosion-fatigue crack propagation.](image)

3 Case Study
3.1 Synthetic Aircraft Fleet Data

Consider a hypothetical control point on the underside of an aircraft wing, as illustrated in Figure 5. For simplicity, we consider the aircraft fleet can fly the 10 different flight types detailed in Table 1. Each flight type is characterized by a load frequency distribution. Therefore, these flights present different levels of severity in terms of mechanical loads. In order to balance out the exposure of the fleet to the severe flights while attending their demands, airline companies rotate their aircraft through different mission mixes (combinations of route structures mixing different flight types). Here, we considered the 10 arbitrarily designed mission mixes. In our study, each one of the 150 aircraft of the fleet is assigned to only one specific mission mix. Within a given mission mix, flights types are assigned to the aircraft randomly following the probability distribution detailed in Table 2. For examples, aircraft belonging to mission mix #5 will never fly flight types I and II while they will fly flight types VII and VIII 4.5% and 10.5% of the time, respectively. In other words, when we assign a flight type to an aircraft, we are imposing a loading frequency as detailed in Table 1. Based on the load magnitude each loading cycle of the flight can be translated into far-field stress (\( \Delta S = S_{max} - S_{min} \)) and stress ratio (\( R = S_{min}/S_{max} \)) values.

Original equipment manufacturers, airline companies, and service providers usually maintain large aircraft fleets (in the range of hundreds to thousands of aircraft). Here, we consider a fleet of 150 aircraft, which are scheduled to fly 8 flights per day. The control points of interest, illustrated by Figure 5, is considered to be on panels made of Al 2024-T3 alloy (coefficients for crack propagation in pure air known and shown in Figure 1a) with \( a_0 = 0.5 \) (mm) and \( a_{max} = 20 \) (mm). Finally, we assume that inspection of this control point is part of the scheduled maintenance. Here, we arbitrarily consider that the first inspection is available for only part of the fleet after 5 years of operation (i.e., nearly after 15,000 flights).

![Fig. 5: Control point on the underside of the aircraft wing.](image)
Table 1: Flight type load distribution and related minimum and maximum stresses (MPa) (adapted from [62]).

<table>
<thead>
<tr>
<th>Load case</th>
<th>Smin</th>
<th>Smax</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
<th>V</th>
<th>VI</th>
<th>VII</th>
<th>VIII</th>
<th>IX</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.07</td>
<td>30.5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>2.45</td>
<td>28</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>4.8</td>
<td>26</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>7.15</td>
<td>23</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>9.5</td>
<td>21</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>11.85</td>
<td>19</td>
<td>30</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2: Mission mix configuration (flight type probability per mission).

<table>
<thead>
<tr>
<th>Mix</th>
<th>I</th>
<th>II</th>
<th>III</th>
<th>IV</th>
<th>V</th>
<th>VI</th>
<th>VII</th>
<th>VIII</th>
<th>IX</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.10⁻⁵</td>
<td>7.5 10⁻⁵</td>
<td>1.10⁻³</td>
<td>2.4 10⁻³</td>
<td>1.1 10⁻²</td>
<td>0.01</td>
<td>0.095</td>
<td>0.15</td>
<td>0.248</td>
<td>0.4824</td>
</tr>
<tr>
<td>2</td>
<td>2.5 10⁻⁵</td>
<td>2.5 10⁻⁵</td>
<td>1.2 10⁻³</td>
<td>1.5 10⁻³</td>
<td>6.75  10⁻³</td>
<td>0.025</td>
<td>0.035</td>
<td>0.155</td>
<td>0.273</td>
<td>0.5025</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1.3 10⁻³</td>
<td>2.3 10⁻³</td>
<td>1.6 10⁻²</td>
<td>0.025</td>
<td>0.025</td>
<td>0.105</td>
<td>0.273</td>
<td>0.5530</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5.10⁻⁵</td>
<td>9.5 10⁻⁴</td>
<td>2.5 10⁻³</td>
<td>8.5 10⁻³</td>
<td>0.013</td>
<td>0.095</td>
<td>0.055</td>
<td>0.323</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1.2 10⁻³</td>
<td>2.3 10⁻³</td>
<td>6.10⁻³</td>
<td>0.015</td>
<td>0.045</td>
<td>0.105</td>
<td>0.2975</td>
<td>0.5278</td>
</tr>
<tr>
<td>6</td>
<td>1.10⁻⁴</td>
<td>2.5 10⁻⁵</td>
<td>7.5 10⁻⁴</td>
<td>2.3 10⁻³</td>
<td>6.13 10⁻³</td>
<td>0.015</td>
<td>0.045</td>
<td>0.08</td>
<td>0.3975</td>
<td>0.4528</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.3 10⁻³</td>
<td>7.3 10⁻³</td>
<td>0.015</td>
<td>0.070</td>
<td>0.055</td>
<td>0.3225</td>
<td>0.5278</td>
</tr>
<tr>
<td>8</td>
<td>2.5 10⁻⁵</td>
<td>0</td>
<td>1.10⁻³</td>
<td>2.5 10⁻³</td>
<td>1.10⁻³</td>
<td>0.02</td>
<td>0.02</td>
<td>0.12</td>
<td>0.285</td>
<td>0.5528</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>7.5 10⁻⁵</td>
<td>7.5 10⁻⁴</td>
<td>7.7 10⁻³</td>
<td>1.1 10⁻²</td>
<td>0.025</td>
<td>0.025</td>
<td>0.055</td>
<td>0.2225</td>
<td>0.6528</td>
</tr>
<tr>
<td>10</td>
<td>7.5 10⁻⁵</td>
<td>0</td>
<td>7.5 10⁻⁴</td>
<td>2.4 10⁻³</td>
<td>6.25 10⁻³</td>
<td>0.005</td>
<td>0.05</td>
<td>0.08</td>
<td>0.2725</td>
<td>0.5828</td>
</tr>
</tbody>
</table>

In this paper, we consider that corrosive cycles represent, on average, 5% of the total number of cycles in one mission (loads of type E and F are related to take-off and landing, respectively, and thus are subjected to corrosion). We then proceed to penalize these cycles by accumulating damage using curves that are between air and NaCl at 3.5% (see Figure 1b). The exact curve will depend on a corrosivity index \( C_{IDX} \) (which essentially controls the interpolation between the two curves). We use the curve for air when \( C_{IDX} = 0 \). We use the curve for NaCl at 3.5% when \( C_{IDX} = 1 \).

To emulate different corrosive environments in the proposed numerical experiment we consider 10 distinct airports. On each flight, we know from which airports the aircraft took off and landed. Thus, we can proceed to use the respective corrosivity indexes presented in Table 3 to penalize the corrosive loading cycles in the flight. While the actual relationship between the corrosivity index and the \( C \) and \( m \) constants is unknown; here, we arbitrarily used the following equations:

\[
C = -2.275 \times 10^{-8} (C_{IDX})^2 + 4.505 \times 10^{-8} C_{IDX} + 1.132 \times 10^{-10}, \text{ and } m = 2.046(C_{IDX})^2 - 4.052 C_{IDX} + 3.859.
\] (8)

As we discussed, Eq. 8 is arbitrarily used in this study as a way to generate synthetic data. In reality, the mapping between corrosivity agents (such as NaCl) and damage accumulation rate (for example, through coefficients of Paris law) is unknown (or only partially understood). The main contribution of the paper is that we use our hybrid physics-informed neural networks to learn that behavior (and adjust the physics-informed portion of the model) with data that could be observed in real life.

Even the few cycles that have the panels exposed to corrosion (average of 5%) can drastically accelerate damage accumulation. When corrosion is not considered, the 5 years inspection is expected to return crack lengths smaller than 1
Table 3: Airport corrosion index values.

<table>
<thead>
<tr>
<th>Airport</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>C^{IDX}</td>
<td>0.517</td>
<td>0.389</td>
<td>0.361</td>
<td>0.545</td>
<td>0.781</td>
<td>1</td>
<td>0.125</td>
<td>0.58</td>
<td>0.608</td>
<td>0.844</td>
</tr>
</tbody>
</table>

(mm). However, in the presence of these few corrosive cycles, the fleet crack envelope is above that value, with few aircraft having crack lengths above the 20 mm mark. We consider that, for every aircraft in the fleet, the far-field cyclic stresses and airport corrosivity index of every flight are available. In real life, the far-field cyclic stresses would be obtained by using engineering models (likely based on finite element analysis) that relate specific maneuvers with cyclic loads (here, we rely on Tables 1 and 2). Corrosivity index could be obtained by analyzing meteorological and environmental data. This information is readily available for most airports, although proprietary at times (here we rely on the arbitrarily chosen values shown in Table 3). In terms of crack length data, we assume that for 10% of the fleet (i.e., 15 aircraft) inspection is performed at the end of the 5th year of operation. Hence, the training set can be summarized as:

1. Observed outputs: crack length at 15 inspected aircraft.
2. Observed inputs: time series for $a_t$, $\Delta S_t$, $R_t$, and $C^{IDX}_t$. The number of cycles depends on the mission mix each aircraft is subject to. By the end of the 5th year, the average number of cycles is around 174,000. Since we used 15 aircraft for training, we actually observed around 2,625,000 input conditions.

3.2 Physics-informed Neural Network Design

With the information discussed in Section 3.1, we proceed to build our hybrid physics-informed neural network model for corrosion-fatigue (illustrated in Figure 4). Table 4 details all the multi-layer perceptron architectures evaluated in this paper. We initially evaluated all the described architectures and, based on their training performance (i.e. best loss function value, and cross-validation error), we selected the best architecture for the damage forecast analysis. We opted for using these architectures to illustrate the ability to fit a neural network with varying number of trainable parameters. No attempt was made to further refine the multi-layer perceptron architecture and its accuracy.

Table 4: Multi-layer perceptron (MLP) configurations used to model the stress intensity range.

<table>
<thead>
<tr>
<th>Layer #</th>
<th>MLP #1</th>
<th>MLP #2</th>
<th>MLP #3</th>
<th>MLP #4</th>
<th>MLP #5</th>
<th>MLP #6</th>
<th>MLP #7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5 / tanh</td>
<td>10 / elu</td>
<td>10 / elu</td>
<td>10 / tanh</td>
<td>20 / tanh</td>
<td>20 / elu</td>
<td>40 / elu</td>
</tr>
<tr>
<td>1</td>
<td>1 / linear</td>
<td>5 / elu</td>
<td>5 / sigmoid</td>
<td>5 / tanh</td>
<td>10 / elu</td>
<td>10 / sigmoid</td>
<td>20 / sigmoid</td>
</tr>
<tr>
<td>2</td>
<td>1 / linear</td>
<td>1 / elu</td>
<td>1 / elu</td>
<td>5 / sigmoid</td>
<td>5 / sigmoid</td>
<td>10 / sigmoid</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 / linear</td>
<td>1 / elu</td>
<td>1 / elu</td>
<td>1 / elu</td>
<td>1 / elu</td>
<td>1 / elu</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>31</td>
<td>111</td>
<td>111</td>
<td>111</td>
<td>371</td>
<td>371</td>
<td>1241</td>
</tr>
</tbody>
</table>

The constructed multi-layer perceptrons take four inputs (crack length, far-field stress, stress ratio and corrosivity index) and provide one output (bias in damage increment due to corrosion). It worth mentioning that the multi-layer perceptron output ($\Delta a_{CORR}$) is hidden, meaning that we do not observe its value directly. We only observe the recurrent neural network output, $a_t$, i.e. the crack length after a given number of flights. We can still build this model since the recurrent neural network performs the damage integration (Figures 2c and 4) adding $a_{t-1}$ to the damage increment contribution due to fatigue $\Delta a_{MECH}$ and the damage increment contribution due to corrosion $\Delta a_{CORR}$. Interestingly, since $\Delta a_{CORR}$ is hidden, it leverages the roughly 2,625,000 observed input conditions during its training. Although models described in Table 4 can have as many as 1,241 trainable parameters, there is no risk of model overfitting.

The optimization of recurrent neural network hyperparameters can be challenging and it is conducted through a gradient-descent algorithm. An initial guess for the hyperparameters that is far away from the optimum might cause divergence of the optimization (or at least a very long time of training). In this paper, we also propose a way to initialize the multi-layer
perceptron parameters. We propose initializing the multi-layer perceptron hyperparameters by using auxiliary planes derived from a simple linear representation of the input-output relationship:

$$\Delta a_{CORR} = \beta_0 + \beta_1 \Delta S + \beta_2 R + \beta_3 C_{IDX} + \beta_4 a$$

(9)

where $\beta_i$ are carefully chosen to be physically reasonable.

In other words, we reckon that practitioners would be able to define rough order of magnitude of damage increment as well as its directionality with respect to the input variables. The coefficients are initialized using engineering judgment. For instance, we know that when the corrosion index is zero the bias is also zero, and it is safe to assume that the bias value increases with far-field stress, corrosivity index and crack length (see Figure 1b). Using these constraints we randomly generated 20 auxiliary planes. This procedure is used only to yield an initial point for the recurrent neural network training and it does not interfere with its overall performance.

The recurrent neural network is fitted with observations for inputs throughout the time series and observations for crack length only at inspection (besides the initial crack length). In other words, the multi-layer perceptron outputs ($\Delta a_{CORR}$) are latent variables of the model. In the training of the recurrent neural network, we use the mean absolute percentage error (MAPE) as loss function:

$$L = MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|a_{\text{PRED},i} - a_{\text{OBS},i}|}{a_{\text{OBS},i}}$$

(10)

where $n$ is the number of observations (inspected aircraft), $a_{\text{PRED},i}$ and $a_{\text{OBS},i}$ are the predicted and observed crack lengths for the $i^{th}$ inspection, respectively. All training sessions were performed as using RMSProp\(^3\) as an optimizer, setup with a learning rate of $10^{-12}$ for 25 epochs.

The interested reader is referred to [63] for a discussion about $MAPE$. Errors of identical magnitude but opposite sign can have different contributions to the $MAPE$. Further, when actual values are small or zero, then contributions to the $MAPE$ are very large. In our application, we do not have problems with the sign of the errors (since $a_{\text{OBS},i} > 0$ and $|a_{\text{PRED},i} - a_{\text{OBS},i}| \geq 0$). Nevertheless, $MAPE$ accentuates the contribution of small crack lengths. Again, in this application, we do not find this to be problematic given the cumulative nature of damage. With slightly estimation of the small cracks in training, we avoid underestimation in damage forecast (as we integrate damage over time).

3.3 Replication of results

Our implementation is all done in TensorFlow\(^4\) (version 2.0.0-beta1) using the Python application programming interface. In order to replicate the results presented here, the interested reader can download the codes and data. First, install the “PINN” python package (base for physics-informed neural networks used in this work) available at [64]. Links for the required data sets and Python scripts demonstrating the proposed framework can be found in [65]. All simulations were conducted using a laptop configured with an Intel Core i7-7820HQ CPU at 2.90GHz, 32GB of RAM, and NVIDIA Quadro M620 graphical processing unit running Windows 10.

4 Results and Discussion

4.1 Recurrent Neural Network Training

As detailed in section 3.1, the proposed physics-informed neural network was trained considering crack length data for 10% of the fleet at the end of the 5th year of operation. Also, as described in section 3.2, different multi-layer perceptron architectures (see Table 4) were evaluated along with distinct auxiliary planes for the multi-layer perceptron hyperparameters initialization. Figure 6a illustrates the loss function history of all multi-layer perceptron architectures, considering the same constraints (i.e. auxiliary plane). The number of epochs that yields convergence for each multi-layer perceptron varies, but a rapid convergence rate is noticeable with most architectures achieving stagnation in 10 epochs or less. On the other hand, Figure 6b shows the loss function history of the multi-layer perceptron architecture #7 across all the 20 auxiliary planes. Even with the engineered multi-layer perceptron weights initialization, convergence to same value of loss function is not guaranteed. Moreover, 3 out of the 20 considered auxiliary planes eventually led to “not a number” loss function values (caused by consecutive very large $\Delta a_{CORR}$), which essentially stops the optimization task. As expected, the success in optimizing the hyperparameters depends on the initial guess. However, we propose different initial guesses following the approach described in section 3.2. With respect to computational cost, the average wall-clock time for training was of 168 min.

\(^3\)https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/RMSprop
\(^4\)https://www.tensorflow.org
Fig. 6: Loss function histories. Multi-layer perceptron architectures are detailed in Table 4.

Table 5 summarizes the optimization performance (for all multi-layer perceptron architectures across the 20 planes used for weight initialization). The conversion ratio is the number of times that the optimization managed to reduce the loss function without leading to ‘non a number’ within the considered number of epochs. All proposed multi-layer perceptron architectures exhibited convergence ratios of at least 75% for the considered initial guesses. This results indicates that the recurrent neural network training is more sensitive to the initial guess than the multi-layer architecture itself. In general, the training sessions in which the recurrent neural network converged were based on the same engineered planes used for initializing the trainable parameters. As expected, Table 5 the more complex architectures (MLP #6 and MLP #7) lead to the lowest loss function values (tending to better represent the input-output relationship in the considered case study when compared to the other architectures).

Table 5: Multi-layer perceptron (MLP) architectures convergence analysis. These results evaluate the training performance of the multi-layer perceptron architectures proposed in Table 4 considering the 20 initial guesses provided by the auxiliary planes. The conversion ratio is the number of times that the optimization successfully finished given the considered number of epochs. The best obtained value of the loss function (Eq. 10) is also presented.

<table>
<thead>
<tr>
<th>MLP</th>
<th>#1</th>
<th>#2</th>
<th>#3</th>
<th>#4</th>
<th>#5</th>
<th>#6</th>
<th>#7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion ratio (out of 20)</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>16</td>
<td>16</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>Best loss value (%)</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>12</td>
<td>14</td>
</tr>
</tbody>
</table>

Next, we study whether cross-validation analysis as presented in [66] can be used to help choosing a multi-layer perceptron architecture amongst the ones considered for this problem. We considered only the best initial guess for each architecture and; to keep the computational cost low, we used 5-fold cross validation. We randomly split our training set into five subsets; and on each iteration of the cross-validation, three aircraft were removed from the training process. After the training is finished, the predictions are checked against the three aircraft left out. Cross-validation errors are defined as $e_{XV} = a_{XV} - a_{OBS}$, where $a_{XV}$ are the cross-validation predicted crack lengths and $a_{OBS}$ are the actual observed crack lengths (at the end of the $5^{th}$ year of operation). Percent cross-validation errors are defined as $\%e_{XV,i} = 100 \times \frac{|e_{XV,i}|}{a_{OBS,i}}$. Figure 7a presents a box plot of the cross-validation absolute errors. The ranking of the models by cross-validation agree with the ranking given by ‘best loss values’ presented in Table 5 (indicating that the more complex architectures in set, MLP #6 and MLP #7, are the most suitable for the considered case study). Figures 7b and 7c illustrate the cross-validation predictions against the actual crack length values. The simpler architectures (MLP #1 to MLP #5) tend to underestimate the crack length values higher than 10 mm, while the complex architectures (MLP #6 to MLP #7) tend to overestimate them. Underestimating the large crack lengths is highly undesirable as it would undermine the safe operation of the fleet. With the cross-validation analysis results we can choose the model that not only leads to the best prediction, but it is also slightly conservative. Assuming that a practitioner wants to avoid underestimation of large crack length values, we opt to select architecture #6 for the remainder of this paper.

5In sequence to sequence modeling, it is also common to find strategies for splitting the data along the time or sequence axis. In our application, for the sake of training our hybrid physics-informed neural networks, only inputs are observed throughout time. The output is known at $t = 0$ and observed at the time of inspection (not the entire history, only the value at end of 5th year). Therefore, any splits along the time or sequence axis are not viable.
Finally, we also studied the contribution of the number of inspected aircraft to the recurrent neural network training. For this analysis we considered MLP architecture #6 with its best initial guess trained with three distinct training set sizes: 5, 10 and 15 inspected aircraft in the 5th year of operation. Figure 8 presents the overall crack length prediction for each training set size after training. One immediate observation, as expected, is that the more aircraft are inspected, the better the recurrent neural network is (predictions are closer to actual values). However, another observation is that with small number of inspections, it is naturally harder to sample the entire range of observed crack lengths, 0 to 25 (mm) As a matter of fact, when 5 aircraft are inspected, the crack length observations are limited to less than 5 (mm). With 10 inspected aircraft there is only one aircraft with crack length between 15 and 20 (mm). The final conclusion is that as much as the number of inspected aircraft contributes to the quality of the resulting model, the coverage of output observation is equally important.

4.2 Corrosion-fatigue Diagnosis and Prognosis

After the recurrent neural network was trained, the resulting model was used to perform crack length estimation at the end of the 5th year of operation (diagnosis) and damage forecast at the end of 6th year of operation (prognosis) across the entire fleet (150 aircraft). Figure 9a presents the results of the crack length estimation at the 5th year of operation for the entire fleet. We can compare the crack lengths predictions of the proposed hybrid recurrent neural network with the ones coming from the fleet (i.e. full corrosion-fatigue model) as well as the purely mechanical fatigue model (i.e. Walker’s equation). The difference between the last two is the bias estimated by the hybrid model. It is noticeable in Figure 9a how the recurrent neural network predictions captures the overall trend of the corrosion-fatigue damage, indicating that the proposed hybrid model is acting as a bias estimator (missing physics). In terms of fleet management, Figure 9a shows that the hybrid recurrent neural network can be used to prioritize which aircraft should be inspected next (the ones with highest predicted crack lengths).

With regards to prognosis, we split the analysis into two time frames: a short-term analysis in which we are forecasting a single year of fleet operation (simulating into the end of the 6th year of operation); and a long-term analysis in which we forecast damage accumulation until the end of the 10th year of operation. Figure 9b presents the results of the short-term analysis showing the predicted versus actual crack lengths at the end of the 6th year of operation. As we discussed before, the model is known to be conservative; and therefore; the predicted values are expected to be higher than the actual ones. In terms
of fleet management, we could use this predictions to evaluate how many aircraft in the fleet would be above an arbitrary repair/replacement threshold. Figure 9b shows that, if this threshold is 20 (mm), we correctly flagged all aircraft with actual crack length above it (22 true positives and no false negatives). In safety critical applications, having no false negatives is extremely important (an aircraft being wrongly cleared to flight can have serious implications). We also see a small number of false positives. These 5 aircraft would be flagged for repair/replacement while in reality at least 2 of those would be very near the threshold of 20 (mm).

Finally, we also evaluate long-term damage forecast (meaning, performing damage estimation up to the 10th year of operation). The main findings of this analysis are represented by the empirical cumulative density function (ECDF) illustrated in Figure 9c. Obtained results confirms that the selected model is conservative. However, in terms of fleet management, it is useful as it warns operators of the rapid increase over time in the probability that the crack will exceed the threshold of 20 (mm). The number grows from around 5% by end of the 5th year to almost 60% by the end of the 7th. This information can guide how fast mitigation strategies have to be put in place.

(a) Recurrent neural network crack length prediction across entire fleet at the end of 5th year of operation.

(b) Recurrent neural network crack length prediction at the end of 6th year of operation. Plot is clipped at 40 (mm) even though predictions can be higher than that.

(c) Probability that the crack length is above 20 (mm), given by the empirical cumulative density function (ECDF), from the 5th to the 10th year of operation.

Fig. 9: Corrosion-fatigue diagnosis and prognosis.

5 Summary and Conclusions

In this contribution, we proposed modeling damage accumulation by simultaneously using physics-informed and data-driven layers within a recurrent neural network. The major achievement of such a hybrid model is the ability to use the data-driven layers to compensate for the missing physics in reduced-order models and accurately estimate damage accumulation. In our numerical experiments, this reduced the need for observing the output of interest at all times. In other words, the physics-informed layers reduce the need for large datasets found in purely data-driven approaches.

In order to evaluate the performance of the proposed framework, we presented a numerical case study focusing on corrosion-fatigue crack propagation. In the considered case, the physics-informed layer only accounts for purely mechanical fatigue. The data-driven layer, on the other hand, adjusts the damage accumulation rate and compensate for the corrosion effects (i.e., working as a bias corrector).

We designed a numerical experiment where we simulated (a) a fleet of 150 aircraft flying different mission mixes (which implies in variation of both mechanical loads and exposure to corrosive environments), and (b) inspection of 10% of the fleet at the end of the 5th year of operation. With the help of this numerical experiment, we have studied:

1. **Initialization of the neural network parameters:** we proposed a simple strategy for initializing the data-driven portion of our hybrid model (a multi-layer perceptron). We argue that engineers and scientists would be able to prescribe the first order effects of input variables in the damage accumulation rate. We observed that when neural network parameters are poorly initialized, their optimization is not guaranteed.

2. **Complexity of neural network architecture and the number of available observations during training:** we performed an empirical study varying the depth and activation functions used in the multi-layer perceptron of our hybrid model. For this particular problem and data set, we found marginal influence of the chosen architecture in the final prediction capability of the recurrent neural network. We believe this is due to the fact that the ratio between available data and number of trainable parameters remained high (i.e., large number of data points compared to hyperparameters) across the multi-layer perceptrons studied here. In training, we observed data for 15 aircraft flying on average 174,000 cycles
(around 2,610,000 observed input conditions) while the number of trainable parameters in the considered multi-layer perceptrons ranged from 31 to 1,241.

3. **Use of the hybrid models for diagnosis and prognosis**: we used the obtained model to predict damage across the fleet close (short-term analysis) and further (long-term analysis) of the inspection point. We learned that the model is able to successfully forecasts damage both in the short-term (one year after partial fleet inspection) and in the long-term (up to five years after partial fleet inspection). The model was slightly conservative, which resulted in manageable number of false positives (aircraft with damage wrongly tagged above an arbitrary threshold) and no false negatives (aircraft mistakenly tagged with damage below the arbitrary threshold).

Despite the many desirable features, the results indicate that the proposed methodology has the following deficiencies (potential topics for future research):

1. **High sensitivity of training results with respect to the initial neural network hyperparameters**: even using the proposed auxiliary planes for the neural network initialization, a poorly initialization of such parameters can potentially derail the training. Therefore, optimization with multiple initial hyperparameter values is still needed.

2. **Model predictions do not interpolate crack length at the training (or test/validation) points**: depending on the application, this can lead to either to undesired conservatism or underprediction.

The presented results motivate us to extended the study in several aspects. For example, we suggest studying the effect of improved physics of failure models (e.g., by including pit growth in the cumulative damage). We also suggest addressing multiple sources of uncertainty within the model and proposing ways to handle them using deep neural networks. For example, one can study the uncertainty in the loads model (from cycle counting to estimation of far-field stresses through finite element) and/or the scatter in material properties (including uncertainty and limitations in coupon data). Finally, one can also study how this physics-informed neural networks could be used to help decision making in fleet management of industrial assets (inspection optimization, fleet recommissioning, and repair/replacement).

**Acknowledgments**

This work was supported by the University of Central Florida (UCF). Nevertheless, any view, opinion, and conclusions expressed in this material are those of the authors alone. Therefore, UCF does not accept any liability in regard thereto.

**References**


