NP-ODE: Neural process aided ordinary differential equations for uncertainty quantification of finite element analysis

Yinan Wang, Kaiwen Wang, Wenjun Cai & Xiaowei Yue

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2021.1891485

NP-ODE: Neural process aided ordinary differential equations for uncertainty quantification of finite element analysis

Yinan Wang[a], Kaiwen Wang[b], Wenjun Cai[b], and Xiaowei Yue[a]

[a]Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA;
[b]Materials Science and Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

ABSTRACT
Finite Element Analysis (FEA) has been widely used to generate simulations of complex nonlinear systems. Despite its strength and accuracy, FEA usually has two limitations: (i) running high-fidelity FEA often requires high computational cost and consumes a large amount of time; (ii) FEA is a deterministic method that is insufficient for uncertainty quantification when modeling complex systems with various types of uncertainties. In this article, a physics-informed data-driven surrogate model, named Neural Process Aided Ordinary Differential Equation (NP-ODE), is proposed to model the FEA simulations and capture both input and output uncertainties. To validate the advantages of the proposed NP-ODE, we conduct experiments on both the simulation data generated from a given ordinary differential equation and the data collected from a real FEA platform for tribocorrosion. The results show that the proposed NP-ODE outperforms benchmark methods. The NP-ODE method realizes the smallest predictive error as well as generating the most reasonable confidence intervals with the best coverage on testing data points. Appendices, code, and data are available in the supplementary files.

ARTICLE HISTORY
Received 3 September 2020
Accepted 26 January 2021

KEYWORDS
Finite element analysis; neural network; neural processes; ordinary differential equation; surrogate model; uncertainty quantification

1. Introduction

1.1. Motivation

Finite Element Analysis (FEA) is a powerful tool to simulate physical phenomena or operations; it is a numerical method that can be used to solve differential equations. Typically, to analyze a large and complex system, the FEA method first divides the entire system into small pieces called finite elements, which is achieved by constructing a mesh of the object. The equations that describe each finite element are then assembled to model the entire system. The approximated solution of the entire system can be finally generated by solving differential equations of all finite elements under environments with a combination of loads and constraints. As FEA provides outstanding accuracy when modeling complex nonlinear systems, it has been applied to many fields, including advanced manufacturing, new material design, heat transfer, solid/fluid mechanics, and multiphysics systems (Zienkiewicz *et al.*, 2013).

Despite the strength of the FEA method in accurately analyzing complex systems, the application of FEA is hindered by several limitations, which can be summarized into two main areas. On the one hand, a high-fidelity FEA simulation usually incurs a high computational cost (Su *et al.*, 2017; Wang, Chen, Kang, Deng, and Jin, 2020). This issue can be critical in some scenarios, such as using ultra-fine elements in FEA, repeating simulations for the function validation, etc. On the other hand, FEA is a deterministic simulation that does not explicitly consider system uncertainties. For example, intrinsic uncertainty can come from input measurement errors or parameter setting deviations, and extrinsic uncertainty may be caused by computational and measurement errors for responses. In most systems with complicated nonlinear behaviors, it is a challenge to obtain deterministic and accurate measurements of the parameters when building the FEA, thus uncertainty quantification is in high demand in FEA studies (Mahadevan and Liang, 2011; Wang *et al.*, 2021).

1.2. Literature review

Recent literature has proposed many methods to tackle the aforementioned limitations of FEA, which can be divided into two branches. One branch of research deals with high computational costs by using surrogate models to approximate and replace the FEA in the system modeling. The other branch takes the system uncertainties into modeling efforts by either directly designing stochastic simulation methods or building stochastic surrogate models to replace the FEA method. A detailed review of these two branches of work is introduced as follows.

1.2.1. Review on deterministic surrogates

Data-driven deterministic surrogates are commonly used to approximate and replace FEA. The basic idea is to train and validate the data-driven surrogate model based on the inputs

CONTACT Xiaowei Yue xwy@vt.edu; Wenjun Cai caiw@vt.edu
Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2021.1891485

and outputs of the FEA, where uncertainties in the simulation output given the same input are ignored. As long as the trained surrogate is accurate, it can bypass the FEA calculation process and replace the FEA in further applications. Kriging or Gaussian Process (GP) regressions are widely used as surrogates. Yue and Shi (2018) proposed a surrogate model-based optimal feed-forward control strategy using a universal kriging model for dimensional variation reduction and defect prevention in the assembly process of composite structures. Wang, Chen, Kang, Deng, and Jin (2020) proposed a GP-constrained general path model to approximate the high-fidelity FEA for efficient product and process design in additive manufacturing. The Deep Neural Network (DNN) is another type of surrogate. Dong *et al.* (2020) proposed a systematic implementation of the Artificial Neural Network to replace FEA in the structural optimization of elastic metamaterials. Liang *et al.* (2018) developed a deep learning model to replace structural FEA in estimating the stress distributions of the aorta. Although the proposed data-driven surrogates provide an efficient alternative to FEA, a common limitation among these pure data-driven models is the lack of physical insight.

To overcome the limitation of pure data-driven surrogates, physics-informed data-driven surrogates have been proposed in the machine learning domain; these do incorporate physical insights into surrogate design. Loose *et al.* (2009) derived one physical-analysis-driven surrogate model based on first principles. Considering that most physical phenomena can be modeled by differential equations, applying a Neural Network (NN) to solve differential equations has been researched. Chen *et al.* (2018) proposed the Neural Ordinary Differential Equation (Neural-ODE) approach, which applied neural networks to parameterize the derivatives of the underlying function between the input and the output, rather than directly modeling their mapping. Yildiz *et al.* (2019) proposed a latent second-order ODE model for high-dimensional sequential data, which explicitly decomposed the latent space into momentum and position components. The Neural-ODE approach has shown its strength in analyzing systems that are governed by differential equations, which may have a great potential to be a suitable surrogate model of FEA.

1.2.2. Review on uncertainty quantification and stochastic surrogates

Researchers have proposed stochastic surrogates by incorporating system uncertainties into computer simulations. Uncertainties inevitably exist in engineering systems, and may be caused by sensing errors, actuating errors, computational errors, etc. FEA simulation is a deterministic method that cannot capture the full effects of system uncertainties. To tackle this issue, Monte Carlo simulation is widely used to capture the uncertainties in the simulation (Rubinstein and Kroese, 2017). Intuitively, this method repeats simulations multiple times and assumes different values of the unknown parameters. Thus, the computational cost grows significantly when the system has a large number of unknown parameters.

Similarly, stochastic surrogates have been proposed to consider the system uncertainties as well as reduce the computational cost (Wang and Shan, 2006). One category of stochastic surrogates is GP-based models. Ankenman *et al.* (2010) extended the kriging to a stochastic simulation setting and proposed stochastic kriging for simulation metamodeling, which characterized both the intrinsic uncertainty inherent in stochastic simulation and the extrinsic uncertainty about the unknown response surface. A surrogate model considering diverse uncertainty sources has been proposed to achieve a better predictive assembly of composite aircraft (Yue *et al.*, 2018). Corresponding active learning has also been developed for maximizing information acquisition with limited samples (Yue et al., 2020). There are some GP-based methods that specifically consider intrinsic (input) uncertainty. Wang *et al.* (2019) refined GP-based optimization algorithms to solve the stochastic simulation optimization problems considering input uncertainty. Wang, Yue, Haaland, and Wu (2020) investigated GP regression considering the input location error within FEA simulations. GP-based methods have shown their advantages as the stochastic surrogate, but they have two limitations: (i) the choice of covariance kernel significantly influences the model performance; and (ii) they are computationally prohibitive for large and high-dimensional datasets (Snelson and Ghahramani, 2006). Hoang *et al.* (2015) proposed the stochastic variational GP to train a sparse GP with a small subset of training data. This method can improve the scalability of GP-based methods in dealing with "Big Data." However, approximated representation in sparse GP has a non-negligible influence on model accuracy, and the parameters introduced by variational inference require extra training time (Liu *et al.*, 2020).

Another category of stochastic surrogates is the extension of DNNs. Blundell *et al.* (2015) proposed a Bayesian Neural Network (BNN) to model uncertainty by learning a distribution of model weights instead of specific values. Gal and Ghahramani (2016) proposed a NN with dropout to capture the variations in training data and evaluate the uncertainties in new simulations. The NN-based stochastic surrogates have shown their strength in modeling large datasets. However, the model performance is highly related to the choice of hyperparameters, such as the prior distribution of model weights and the dropout ratio. Furthermore, learning posterior distributions in a BNN can be difficult considering the high dimensionality and complexity.

Inspired by the idea of GP, Garnelo, Rosenbaum, *et al.* (2018) proposed Conditional Neural Processes (CNPs) to combine NNs with the features of a GP. This method is to define conditional distributions over possible functions given a set of observations. The CNPs can generate predictive results and evaluate the output uncertainties. More general models, Neural Processes (NPs), were proposed to add a latent encoding branch to project the input uncertainties into global latent representations and then incorporate these representations into the generation process of output distributions (Garnelo, Schwarz *et al.*, 2018). Kim *et al.* (2019) integrated the attention module (Bahdanau *et al.*, 2015) into NPs and proposed Attentive

Neural Processes. The attention module can be regarded as a measure of similarity among context inputs (observed) and target inputs (unobserved) to find which context is most relevant to a given target. Overall, the NPs can characterize the input and output uncertainties. However, these methods cannot capture the differential equations structure of FEA, which limits their interpretability.

1.3. Proposed method and contributions

Based on the literature review, there is a need to develop a stochastic surrogate with physical insights so as to reduce the computation cost of the FEA method and capture system uncertainties. To close this research gap, we propose a novel method, Neural Process Aided Ordinary Differential Equations (NP-ODE), to build a physics-informed data-driven surrogate for FEA simulations with uncertainty quantification. The structure of NP-ODE follows the encoder–decoder format, in which the encoder part projects the observed simulations into feature space, and the decoder part takes the features and unobserved query as the input to learn distributions over the output. Compared with a pure data-driven decoder, the NP-ODE method incorporates the Neural-ODE as the decoder to strengthen the model in solving complex systems governed by differential equations. To this extent, both the FEA method and our proposed NP-ODE are built to approximate solutions for systems with underlying differential equations. Thus, incorporating Neural-ODE as the decoder makes the model a convincing surrogate for FEA. What is more, a similar structure to the NPs can enable uncertainty quantification in our proposed NP-ODE. The pipeline of our proposed method is shown in Figure 1. First, an FEA simulation platform is built based on the system's properties and parameters. The FEA platform is further validated by real experimental data. Given the generated simulations from FEA, our proposed method NP-ODE is trained and tested on the datasets to generate the predictive output and conduct uncertainty quantification. Finally, the NP-ODE is ready to be used as a stochastic surrogate of FEA for new inputs.

The contributions of this article can be summarized as follows:

1. The NP-ODE evaluates both the input and output uncertainties and generates distributions over the output to enable uncertainty quantification.
2. Compared with the pure data-driven surrogate, the NP-ODE solves differential equations in its decoders, so it is more physically close to the original FEA, and has better interpretability.
3. Compared with the original decoder in NPs, NP-ODE can reduce the number of parameters by incorporating Neural-ODE to mitigate the risk of overfitting with scarce training samples.
4. The NP-ODE can improve the robustness in dealing with noisy data points governed by the differential equation.

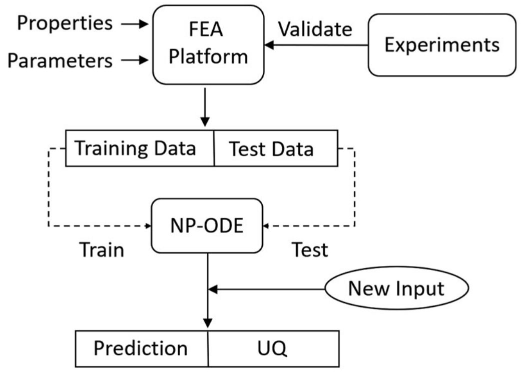


Figure 1. Overview of building Neural-ODE from FEA simulations.

The remainder of this article is organized as follows: Section 2 introduces the mathematical foundations of Neural-ODE and NPs; Section 3 proposes the NP-ODE, investigates the uncertainty quantification and its properties, and develops a computational algorithm for the proposed NP-ODE; Section 4 presents a simulation study as a proof-of-concept; Section 5 presents the case study of FEA simulation for tribocorrosion, and compares the proposed method with benchmark methods; Section 6 gives a brief conclusion on our proposed method and its advantages.

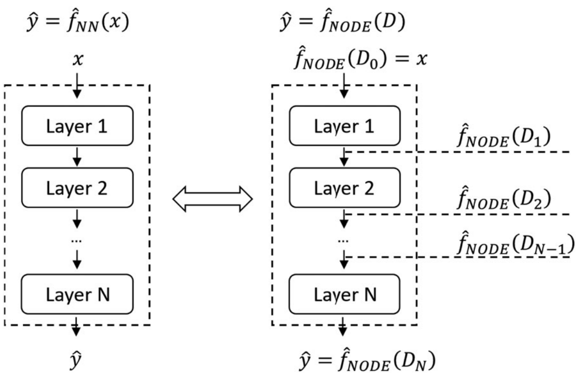
2. Neural ODE and NPs

In this section, we first introduce the Neural-ODE and NPs. The notations that appear in this article are summarized in the supplementary Appendix I.

2.1. Neural-ODE

The Neural-ODE was proposed to parameterize the derivative of a hidden state using a NN (Chen *et al.*, 2018). The Neural-ODE has shown particular strengths in image classification, modeling continuous flows, and modeling time-series data.

We take FEA simulation as one example. For simplicity, suppose both the input and output of FEA are scalars, and FEA simulations generate N data points $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ with the underlying functional relationship $y = f(x),\ x \in \mathbb{R},\ y \in \mathbb{R}$. The classic NN is built to approximate the functional relationship by minimizing a loss function $\mathcal{L}\left(\hat{f}_{NN}(x), y\right)$, in which $\hat{f}_{NN}(x)$ is the approximated mapping given by the NN. As shown in Figure 2, considering the multi-layer structure of NN, $\hat{f}_{NN}(x)$ can be regarded as a series of discrete transformations with input x as the initial value, and the steps of transformations are denoted as the model depth. In this case, the $\hat{f}_{NN}(x)$ is equivalent to $\hat{f}_{NODE}(D)$, $\hat{f}_{NODE}(D_0) = x$, in which $\hat{f}_{NODE}(D)$ represents the discrete transformation with respect to (w.r.t) model depth D, and D_0 represents the initial layer. Instead of directly modeling the functional relationship as discrete transformations, the Neural-ODE is built to approximate the first-order derivative of $\hat{f}_{NODE}(D)$, with the assumption that $\hat{f}_{NODE}(D)$ is continuous and differentiable.



$\hat{y} = \hat{f}_{NN}(x)$
x
Layer 1
Layer 2
Layer N
$\hat{y}$
$\hat{y} = \hat{f}_{NODE}(D)$
$\hat{f}_{NODE}(D_0) = x$
$\hat{f}_{NODE}(D_1)$
$\hat{f}_{NODE}(D_2)$
$\hat{f}_{NODE}(D_{N-1})$
$\hat{y} = \hat{f}_{NODE}(D_N)$

Figure 2. Equivalence between $\hat{f}_{NN}(x)$ and $\hat{f}_{NODE}(D)$.

The model depth D is optimized during the training phase:

$$g_{NODE}(D) = \frac{d\hat{f}_{NODE}(D)}{dD}, \tag{1}$$

in which, the $g_{NODE}(D)$ denotes the first-order derivative given by Neural-ODE. Based on the approximated first-order derivative of $\hat{f}_{NODE}(D)$, the output can be calculated by the ODE solver and is shown in Equation (2):

$$\hat{f}_{NODE}(D_N) = \text{ODESolve}\left(\hat{f}_{NODE}(D_0),\ g_{NODE}(D), D_0, D_N\right). \tag{2}$$

2.2. NPs

The NPs are inspired by the ideas of the GP and designed for modeling the regression function, such as $\mathbf{y} = \mathbf{f}(\mathbf{x})$ between input $\mathbf{x} \in \mathbb{R}^m$ and output $\mathbf{y} \in \mathbb{R}^p$ (Kim *et al.*, 2019). NPs can output distributions over unobserved target FEA data points $(\mathbf{x}_{n+1}, \mathbf{y}_{n+1}), \ldots, (\mathbf{x}_{n+T}, \mathbf{y}_{n+T})$ conditioned on the observed FEA simulations $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$. The model structure of NPs is shown in Figure 3, in which the output of NPs, $\hat{\mathbf{y}}_{n+1} \sim \mathcal{N}(\overline{\mathbf{y}}_{n+1}, \boldsymbol{\sigma}_{n+1})$, is expressed as the posterior distribution given observed data points as shown in Equation (3) (Kim *et al.*, 2019):

$$p(\hat{\mathbf{y}}_{n+1}|\mathbf{x}_{n+1}, (\mathbf{x}_{1:n}, \mathbf{y}_{1:n})) = \int p(\hat{\mathbf{y}}_{n+1}|\mathbf{x}_{n+1}, \mathbf{d}_C, \mathbf{z}) q(\mathbf{z}|\mathbf{s}_C) d\mathbf{z}. \tag{3}$$

In Equation (3), the $q(\mathbf{z}|\mathbf{s}_C,\ \mathbf{z} \in \mathbb{R}^N,\ \mathbf{s}_C \in \mathbb{R}^N)$ is the prior distribution on $\mathbf{z}$ given by the stochastic encoder, $\mathbf{d}_C \in \mathbb{R}^N$ is given by the deterministic encoder, and the likelihood function $p(\hat{\mathbf{y}}_{n+1}|\mathbf{x}_{n+1}, \mathbf{d}_C, \mathbf{z})$ is represented by the decoder. In FEA simulation, data points $(\mathbf{x}_{n+1}, \mathbf{y}_{n+1}), \ldots, (\mathbf{x}_{n+T}, \mathbf{y}_{n+T})$ are from different replications and do not have temporal dependencies or a specific order. Equation (3) shows how to predict one unobserved data point $(\mathbf{x}_{n+1},\ \hat{\mathbf{y}}_{n+1})$.

In the encoder part of NPs, the basic building block is the Multi-Layer Perceptron (MLP). First, the deterministic encoder captures the interactions among observed FEA data and outputs the finite-dimensional representation $\mathbf{d}_C$. The attention module is an important part to aggregate the representations $(\mathbf{d}_1, \ldots, \mathbf{d}_n)$ from each observed data point corresponding to a specific unobserved target simulation point $\mathbf{x}_{n+1}$. Intuitively, the attention module is to compute the weights of each observed keys $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ w.r.t. the unobserved query $\mathbf{x}_{n+1}$, and apply these weights to compute the weighted sum of representations $(\mathbf{d}_1, \ldots, \mathbf{d}_n)$ to generate the output $\mathbf{d}_C$ (Bahdanau *et al.*, 2015). Second, the stochastic encoder will capture the uncertainties in the predictive value $\hat{\mathbf{y}}_{n+1}$ and output the prior distribution $q(\mathbf{z}|\mathbf{s}_C)$ on the latent representation $\mathbf{z}$. As the stochastic representation $\mathbf{s}_C$ of the context data points are generated by the mean aggregation of $(\mathbf{s}_1, \ldots,\ \mathbf{s}_n)$, the prior distribution $q(\mathbf{z}|\mathbf{s}_C)$ is invariant to permutations of context data points. The latent representation $\mathbf{z}$ can account for the uncertainties in predicting $\hat{\mathbf{y}}_{n+1}$. The decoder will take the unobserved query $\mathbf{x}_{n+1}$, deterministic representation $\mathbf{d}_C$ of observed data w.r.t. $\mathbf{x}_{n+1}$, and stochastic representation $\mathbf{s}_C$ of observed data as the input, and calculate the mean $\overline{\mathbf{y}}_{n+1}$ and standard deviation $\boldsymbol{\sigma}_{n+1}$ for the predictive distribution over the target value $\hat{\mathbf{y}}_{n+1}$.

To learn the parameters of NPs, the loss function is defined by maximizing the evidence lower bound based on the historical FEA simulation data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$ and unobserved target simulation $(\mathbf{x}_{n+1}, \mathbf{y}_{n+1}), \ldots, (\mathbf{x}_{n+T}, \mathbf{y}_{n+T})$, as shown in Equation (4) (Kim *et al.*, 2019):

$$\log p(\mathbf{y}_{n+1:n+T}|\mathbf{x}_{n+1:n+T}, (\mathbf{x}_{1:n}, \mathbf{y}_{1:n})) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{s}_T)}\left[\sum_{i=1}^{T} \log p(\mathbf{y}_{n+i}|\mathbf{x}_{n+i}, \mathbf{d}_C, \mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\mathbf{s}_T)\|q(\mathbf{z}|\mathbf{s}_C)). \tag{4}$$

In Equation (4), $q(\mathbf{z}|\mathbf{s}_T)$ represents the posterior distribution on latent representation $\mathbf{z}$ whereas $q(\mathbf{z}|\mathbf{s}_C)$ represents the prior on $\mathbf{z}$; $\log p(\mathbf{y}_{n+i}|\mathbf{x}_{n+i}, \mathbf{d}_C, \mathbf{z})$ represents the log probability of true target value on the predictive likelihood; $\mathbf{s}_T$ represents the stochastic representation with observed and unobserved data points as the input (in the training phase). The detailed derivation of Equation (4) is in the supplementary Appendix II.

3. NP-ODE

In this section, first, the limitations of Neural-ODE and NPs will be clarified. Then, a novel method, NP-ODE, is proposed for predictive analytics and uncertainty quantification of FEA simulations. The properties of the proposed NP-ODE will be discussed. Furthermore, the algorithm of the proposed NP-ODE is developed.

3.1. Limitations of Neural-ODE and NPs

Neural-ODE focuses on using the NN to parameterize the derivative of the hidden state. In this way, the output of the Neural-ODE can be regarded as the integration of inputs over the latent space. Although Neural-ODE is a promising surrogate model of systems governed by differential equations, it is a deterministic model without the capacity for uncertainty quantification. Since uncertainties usually exist in engineering systems, the deterministic property of

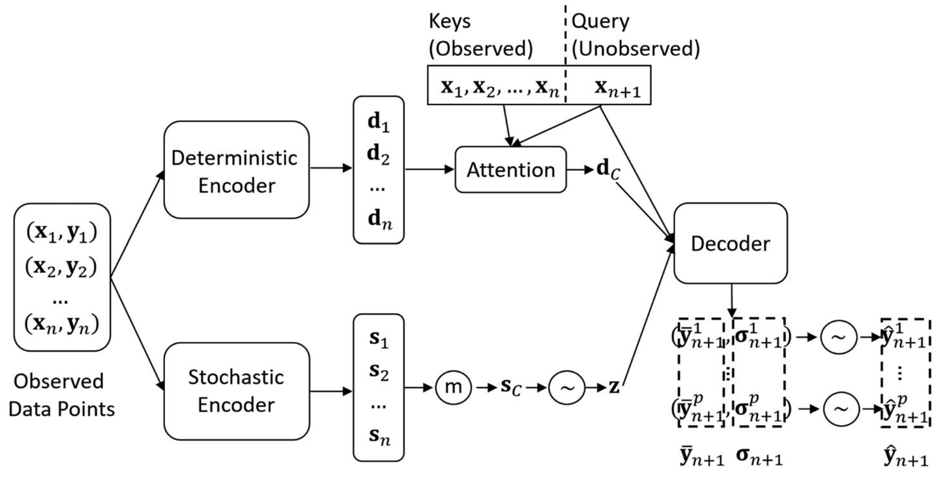

Keys
(Observed)
Query
(Unobserved)
$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$
$\mathbf{x}_{n+1}$
Deterministic Encoder
Stochastic Encoder
Attention
Decoder
$\mathbf{d}_1$ $\mathbf{d}_2$ … $\mathbf{d}_n$
$\mathbf{d}_C$
$\mathbf{s}_1$ $\mathbf{s}_2$ … $\mathbf{s}_n$
m
$\mathbf{s}_C$
$\mathbf{z}$
$(\mathbf{x}_1, \mathbf{y}_1)$ $(\mathbf{x}_2, \mathbf{y}_2)$ … $(\mathbf{x}_n, \mathbf{y}_n)$
Observed Data Points
$\bar{\mathbf{y}}_{n+1}^1$ $\boldsymbol{\sigma}_{n+1}^1$ $\hat{\mathbf{y}}_{n+1}^1$
$\bar{\mathbf{y}}_{n+1}^p$ $\boldsymbol{\sigma}_{n+1}^p$ $\hat{\mathbf{y}}_{n+1}^p$
$\bar{\mathbf{y}}_{n+1}$ $\boldsymbol{\sigma}_{n+1}$ $\hat{\mathbf{y}}_{n+1}$

Figure 3. Model structure of NPs.

Neural-ODE hinders its ability to model FEA simulations and capturing the system uncertainties.

NPs have many advantages, such as fitting observed data efficiently, learning predictive distributions instead of a deterministic function, and generating the mean $\bar{\mathbf{y}}_{n+1}$ and standard deviation $\boldsymbol{\sigma}_{n+1}$ to enable uncertainty quantification of predicted $\hat{\mathbf{y}}_{n+1}$. However, NPs use MLPs as the basic building blocks of its encoder and decoder, which are not the ideal choice, especially in modeling FEA simulations. The reasons can be summarized into three aspects:

1. The model complexity tends to be significant if we have an extensive feature vector. Suppose MLP stacks D Fully Connected (FC) layers, and the feature dimension of each layer is N. Thus, the parameter size for this MLP is DN^2. The number of parameters in the NPs will increase significantly with the higher dimension feature vector.
2. The FEA simulation dataset is usually scarce, because FEA is time-consuming and requires high computational cost. In this situation, applying the model with a large number of parameters to model a scarce dataset may have overfitting issues.
3. The FEA is a numerical method to solve PDEs/ODEs. The MLP is a pure data-driven method that may lack the physical insights concerning differential equations.

To tackle these limitations, we proposed a novel method, NP-ODE, to build an FEA-informed data-driven stochastic surrogate model. Compared with Neural-ODE, our proposed NP-ODE follows the prototype of NPs to capture the uncertainties of predictive results. Compared with NPs, the NP-ODE incorporates Neural-ODE as the decoder, which can reduce the number of parameters and provide physical insights concerning differential equations to the surrogates of FEA simulations.

3.2. General setup of the NP-ODE to model FEA simulations

In this section, we first formulate the Neural-ODE to model FEA simulations. The basic structure of the encoder and decoder in NP-ODE is then illustrated and discussed.

3.2.1. Formulate Neural-ODE to model FEA simulations

FEA is a numerical technique to approximate Partial Differential Equations or Ordinary Differential Equations (PDEs/ODEs), which are often used to describe a complex system. As mentioned above, Neural-ODE is a powerful tool in solving systems governed by differential equations, so it has the potential to be an accurate surrogate for FEA methods. The FEA simulations can be formulated as a multivariate regression in which the input $\mathbf{x} \in \mathbb{R}^m$ and the output $\mathbf{y} \in \mathbb{R}^p$ variables of FEA are the input and the output of the regression, respectively. Instead of directly modeling the mapping between $\mathbf{x}$ and $\mathbf{y}$, the Neural-ODE generates the predicted output by modeling and solving the ODE of the underlying function between $\mathbf{x}$ and $\mathbf{y}$. Suppose the underlying function is $\mathbf{y} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^p$, the Neural-ODE will act as a feature extractor, in which the input $\mathbf{x} \in \mathbb{R}^m$ are firstly mapped into the feature space $\mathbf{w} \in \mathbb{R}^N$, then the continuous transformation is taken over the feature space, and the output $\mathbf{y}$ is given from an extra layer. We select Euler's method as the ODE solver (Griffiths and Higham, 2011). The feature transformation is given in Equations (5):

$$\begin{gathered} D_i = D_{i-1} + \Delta D \\ \mathbf{g}_{NODE}(D_{i-1}) = \left.\frac{d\hat{\mathbf{f}}_{NODE}(D)}{dD}\right|_{D=D_{i-1}} \\ \hat{\mathbf{f}}_{NODE}(D_i) \approx \hat{\mathbf{f}}_{NODE}(D_{i-1}) + \Delta D \mathbf{g}_{NODE}(D_{i-1}), \qquad i = 1, \dots, n \\ \hat{\mathbf{f}}_{NODE}(D_0) = \mathbf{w} \ , \end{gathered} \tag{5}$$

where $\hat{\mathbf{f}}_{NODE}(D)$ represents the approximated discrete feature transformation given by Neural-ODE, D is equivalent to the depth of NN, ΔD is the step size. Assume $\hat{\mathbf{f}}_{NODE}(D)$ is continuous and differentiable, $\mathbf{g}_{NODE}(D)$ denotes the first-order derivative of $\hat{\mathbf{f}}_{NODE}(D)$. In this case, D is similar to the number of the intermediate layers in the original NN. As shown in Figure 2, given the input $\hat{\mathbf{f}}_{NODE}(D_0)$, the output at model depth D_i is denoted as $\hat{\mathbf{f}}_{NODE}(D_i)$, which can be regarded as a series of discrete transformations. The value of D_n can be optimized in the training process, which means the Neural-

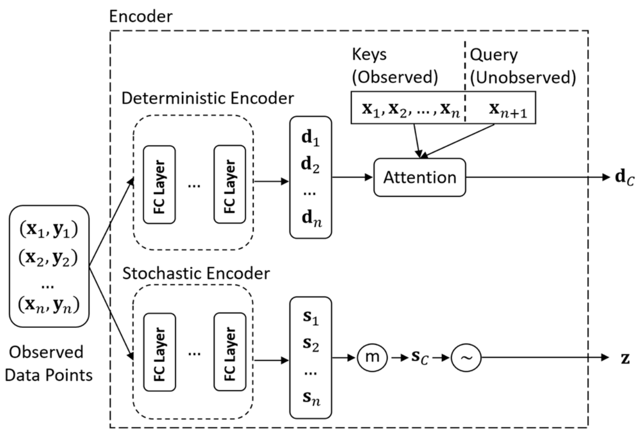


Figure 4. Structure of the encoder in the NP-ODE.

ODE finds the most informative feature in the continuous feature transformation.

In summary, the basic steps of building the Neural-ODE as a surrogate include: (i) formulate FEA simulations as a regression with input $\mathbf{x} \in \mathbb{R}^m$ and output $\mathbf{y} \in \mathbb{R}^p$; (ii) map the input $\mathbf{x} \in \mathbb{R}^m$ into feature space $\mathbf{w}_{D_0} \in \mathbb{R}^N$; (iii) extract the most informative features from the continuous transformation modeled by Neural-ODE; and (iv) map the extracted features into the output space $\mathbf{y} \in \mathbb{R}^p$.

3.2.2. Structure of the encoder in NP-ODE

As shown in Figure 4, the basic structure of NP-ODE mainly consists of an encoder and a decoder. The role of the encoder is to extract the deterministic and stochastic representations of observed data points. The main components are the deterministic encoder, the stochastic encoder, and the attention module.

The deterministic and stochastic encoders share a similar structure, consisting of a MLP (stacked FC layers). The expressions of the deterministic and stochastic encoders are given in Equations (6):

$$\begin{aligned} \mathbf{d_i} &= < \mathbf{W}_{DE}, (\mathbf{x}_i, \ \mathbf{y}_i) > + \mathbf{b}_{DE} \\ \mathbf{s_i} &= < \mathbf{W}_{SE}, (\mathbf{x}_i, \ \mathbf{y}_i) > + \mathbf{b}_{SE}, \end{aligned} \tag{6}$$

In Equations (6), $\mathbf{W}_{DE}$, $\mathbf{b}_{DE}$ denote the weight matrix and bias in the FC layer of the deterministic encoder, $\mathbf{W}_{SE}$, $\mathbf{b}_{SE}$ denote the weight matrix and bias in the FC layer of the stochastic encoder, $\mathbf{d_i}$, $\mathbf{s_i}$ are the deterministic and stochastic representations of observed data point $(\mathbf{x}_i, \ \mathbf{y}_i)$.

The role of the attention module is to take the weighted aggregation of the deterministic representations $(\mathbf{d}_1, \dots, \mathbf{d}_n)$ based on the similarity of observed inputs $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ and the queried (unobserved) input $\mathbf{x}_{n+1}$. A multi-head attention module is applied in the NP-ODE (Vaswani *et al.* 2017). It is worth noting that the output of the attention module has the permutation invariant property so that the order of deterministic representations $(\mathbf{d}_1, \dots, \mathbf{d}_n)$ and observed inputs $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ does not influence the output $\mathbf{d}_c$. Similarly, the $\mathbf{s}_c$ is generated by the mean aggregation of stochastic representations $(\mathbf{s}_1, \dots, \mathbf{s}_n)$, which also has the permutation invariant property.

3.2.3. Structure of the decoder in NP-ODE

The NP-ODE is built to incorporate the Neural-ODE as the decoder in NPs, which not only strengthens its ability in modeling FEA simulations but also gains the capacity to capture system uncertainties.

As shown in Figure 5, in Neural-ODE, the ODE network along with the ODE solver (e.g., Euler method) can approximate the continuous transformation $\mathbf{f}_{NODE}(D)$ of the input representations, and a smaller step size ΔD can give a better approximation accuracy of FEA simulations. In contrast, the MLP decoder in the NPs stacks multiple FC layers, which can be regarded as the discretization of the continuous transformations. Increasing the number of intermediate layers in the MLP may give a better approximation. However, this will increase the number of parameters significantly. Similar to original NNs, the Neural-ODE has the equivalence property associated with the model depth (number of intermediate layers), which is determined by the step size ΔD. As long as the step size ΔD is small enough and the predicted derivative at each step given by the ODE network is accurate enough, the Neural-ODE is equivalent to an infinite-layer NN. The advantage of Neural-ODE is that it uses the same ODE network to generate the derivative of the underlying function at each step, which reduces the number of parameters compared with MLP.

In general, the expressions of the decoder in NP-ODE are summarized as Equations (7).

$$\begin{aligned} \hat{\mathbf{y}}_{n+1} &\sim \mathcal{N}(\bar{\mathbf{y}}_{n+1}, \ \boldsymbol{\sigma}_{n+1}) \\ \bar{\mathbf{y}}_{n+1} &= h_1\left(\hat{\mathbf{f}}_{NODE}(D_n)\right) \\ \boldsymbol{\sigma}_{n+1} &= h_2\left(\hat{\mathbf{f}}_{NODE}(D_n)\right) \\ \hat{\mathbf{f}}_{NODE}(D_i) &\approx \hat{\mathbf{f}}_{NODE}(D_{i-1}) + \Delta D \mathbf{g}_{NODE}(D_{i-1}), \\ \hat{\mathbf{f}}_{NODE}(D_0) &= \mathbf{w} = (\mathbf{d}_C, \mathbf{z}, \mathbf{x}_{n+1}) \\ \mathbf{g}(D_{i-1}) &= \left.\frac{d\hat{\mathbf{f}}_{NODE}(D)}{dD}\right|_{D=D_{i-1}} \\ D_i &= D_{i-1} + \Delta D, \ \ i = 1, \dots, n \end{aligned} \tag{7}$$

In Equations (7), $h_1(\cdot)$ and $h_2(\cdot)$ represent two FC layers used to map the output features from Neural-ODE to predicted mean $\bar{\mathbf{y}}_{n+1}$ and standard deviation $\boldsymbol{\sigma}_{n+1}$, $\hat{\mathbf{f}}_{NODE}(D_i)$ represents the dynamic feature transformations modeled by Neural-ODE, $\mathbf{d}_C$ and $\mathbf{z}$ remain the same as NPs which are the deterministic and stochastic representations of observed data, respectively. The NP-ODE is designed to model systems governed by the differential equations.

Combine the introductions of the encoder and decoder in Sections 3.2.2 and 3.2.3 and the proposed NP-ODE is ready to use. Before demonstrating the performance of NP-ODE, uncertainty quantification and its properties will be further analyzed.

3.3. Uncertainty quantification

Uncertainty Quantification (UQ) conducted by our proposed NP-ODE is to quantitatively measure the uncertainties of predicted results given new FEA inputs. We take the uncertainties in the FEA for tribocorrosion (introduced in

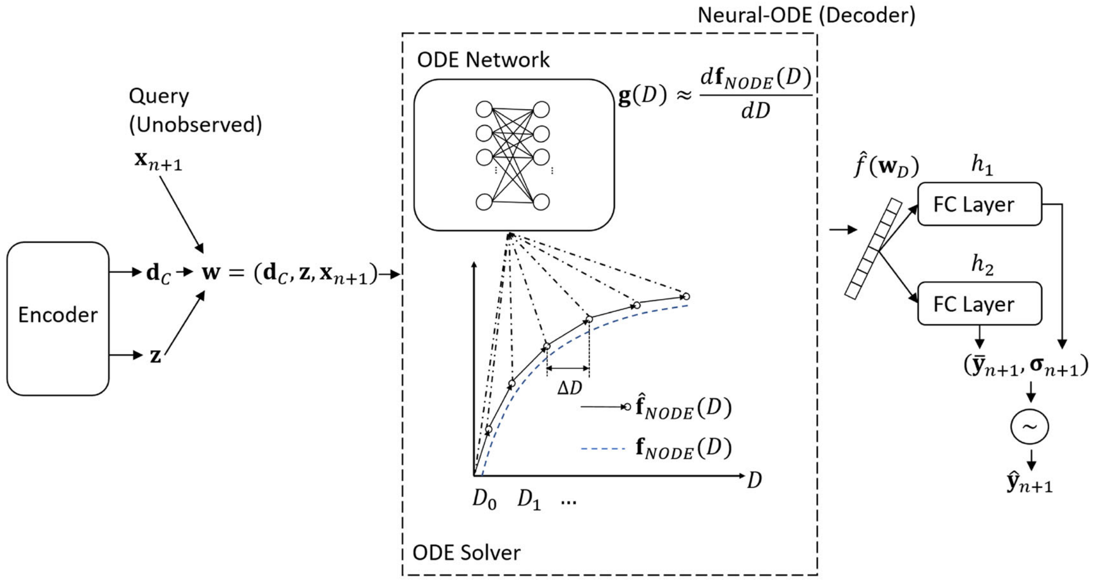


Figure 5. Structure of the decoder in the NP-ODE.

Section 5.1) as an example. There are six parameters representing material properties used as the inputs of FEA. The FEA method for tribocorrosion is built to deterministically simulate the corrosion rate under various combinations of parameters. However, in reality, these parameters are not deterministic in tribocorrosion FEA simulation because: (i) the corrosion process changes the properties of the material; (ii) given the same set of parameters, the corrosion rate (FEA output) may have slight changes in different runs of real experiments. Since the FEA method can only consider a limited number of critical parameters, other environmental parameters may inevitably influence the corrosion rate. The idea of UQ in our proposed NP-ODE lies in capturing the variations of FEA simulations under various combinations of parameters. The output of NP-ODE is a distribution of the predicted result. We select the confidence interval to numerically measure the uncertainties of each output, which means considering the system uncertainties, the corrosion rate is likely to locate within the predicted interval.

3.4. Properties of NP-ODE

The properties of our proposed NP-ODE can be summarized as follows.

First, compared with the MLP decoder, incorporating Neural-ODE as the decoder can reduce the parameter size, since it can eliminate the multi-layer structure. The number of parameters in a D-layer MLP is DN^2, in which N is the feature dimension, D is equivalent to the number of steps in discrete transformations. The number of parameters in the Neural-ODE is not influenced by the number of steps, because the same ODE network will be applied at each step to estimate the derivatives. If the ODE network consists of FC layers, the number of parameters in the Neural-ODE can be expressed as N^2. More importantly, other types of layers (e.g., convolutional (Conv) layer) can also be used as the building block in the ODE network. Compared with the FC layer, the number of parameters in the Conv layer will not be determined by the feature dimension N, which enables the Neural-ODE to further reduce the number of parameters. In our experiments, the Conv layer is selected as the basic building block in the ODE network. A detailed comparison of the number of parameters will be introduced in Section 5.3.

Second, the NP-ODE and the FEA method share the similar idea in emulating complex systems, which is modeling and solving the PDEs/ODEs that represent the complex system, rather than directly modeling the functional relationship between input and output by pure data-driven models.

Third, the proposed NP-ODE captures the output uncertainties and generates the distribution over outputs for FEA simulations. Given the output distribution, we can further conduct UQ on outputs by generating confidence intervals at each data point.

3.5. Pseudo-code of the algorithm for NP-ODE

The pseudo-code of NP-ODE to model FEA simulations and implement uncertainty quantification is summarized in Algorithm 1. $\mathcal{D}_C$ represents the dataset containing inputs and outputs of historical FEA simulations, $\mathcal{D}_T$ represents the dataset containing target data points (unobserved FEA simulation), $\mathcal{X}_T$ represents the dataset containing target FEA inputs.

Algorithm 1: NP-ODE in modeling FEA simulations and conducting UQ

```
Inputs:
1:  $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^p$    ▷ data from FEA simulations
2:  $\mathcal{D}_{train} = (\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$    ▷ training data
    $\mathcal{D}_{test} = (\mathbf{x}_{n+1}, \mathbf{y}_{n+1}), \ldots, (\mathbf{x}_{n+T}, \mathbf{y}_{n+T})$    ▷ testing data
Training:
3:  while $i \le$ number of iterations do
4:      randomly select $\mathcal{D}_C \subseteq \mathcal{D}_{train}$, $\mathcal{D}_T = \mathcal{D}_{train} - \mathcal{D}_C$
5:      prior $q(\mathbf{z}|\mathbf{s}_C) \leftarrow$ feed $\mathcal{D}_C$ into stochastic encoder
6:      posterior $q(\mathbf{z}|\mathbf{s}_T) \leftarrow$ feed $\mathcal{D}_{train}$ into stochastic encoder
7:      deterministic representation $\mathbf{d}_C \leftarrow$ feed $(\mathcal{D}_C, \mathcal{X}_T)$
        into the deterministic encoder
```

```
8:      for $(\mathbf{x}_t,\ \mathbf{y}_t)$ in $\mathcal{D}_T$ do
9:        sample $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{s}_T)$
10:       $\bar{\mathbf{y}}_t, \boldsymbol{\sigma}_t \leftarrow$ feed $(\mathbf{z},\ \mathbf{d}_C,\ \mathbf{x}_t)$ into decoder
11:       $\hat{\mathbf{y}}_t \sim \mathcal{N}(\bar{\mathbf{y}}_t, \boldsymbol{\sigma}_t)$
12:       calculate likelihood $p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{d}_C, \mathbf{z})$
13:     end for
14:     $\mathcal{L} = \mathbb{E}_{q(\mathbf{z}|\mathbf{s}_T)}\left[\sum_{t=1}^{T} \log p(\mathbf{y}_t|\mathbf{x}_t, \mathbf{d}_C, \mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\mathbf{s}_T)\|q(\mathbf{z}|\mathbf{s}_C))$
15:     update parameters to maximize $\mathcal{L}$
16:     $i = i + 1$
17:   end while
Testing:
18:   $\mathcal{D}_C = \mathcal{D}_{train},\ \mathcal{D}_T = \mathcal{D}_{test}$
19:   prior $q(\mathbf{z}|\mathbf{s}_C) \leftarrow$ feed $\mathcal{D}_C$ into stochastic encoder
20:   deterministic representation $\mathbf{d}_C \leftarrow$ feed $(\mathcal{D}_C,\ x_T)$
      into the deterministic encoder
21:   for $(\mathbf{x}_t,\ \mathbf{y}_t)$ in $\mathcal{D}_T$ do
22:     sample $\mathbf{z}$ from $q(\mathbf{z}|\mathbf{s}_C)$
23:     $\bar{\mathbf{y}}_t, \sigma_t \leftarrow$ feed $(\mathbf{z},\ \mathbf{d}_C,\ \mathbf{x}_t)$ into decoder
24:     $\hat{\mathbf{y}}_t \sim \mathcal{N}(\bar{\mathbf{y}}_t, \sigma_t)$
25:     generate confidence interval using predicted distribution
```

In this section, we introduced the general set up of our proposed method NP-ODE along with its expressions, analyzed its properties, and summarized the pseudo-code to illustrate its training and testing procedures. In summary, the advantages of the NP-ODE include (i) compared with NPs, it improves the parameter efficiency to reduce the number of parameters in the decoder; (ii) compared with Neural-ODE, it generates the distribution over the predicted output to enable UQ; (iii) it shares a similar differential equations structure with the FEA method in emulating complex systems. Thus, it is more promising to be a surrogate model of FEA simulations.

4. Simulation study

To validate the effectiveness of our proposed NP-ODE, a simulation study is conducted by restoring dynamic functions using sampled data from a given ODE. Also, to demonstrate the robustness of NP-ODE, the experiment is repetitively conducted on sampled data with different levels of Gaussian random noise. In this section, the simulation setup is first introduced to illustrate how the sampled data is collected. The evaluation metrics are then discussed. Finally, the benchmark method is introduced, and performance comparisons between the NP-ODE and the benchmark are discussed.

4.1. Simulation setup

Two-dimensional spirals are selected to generate simulated data points, which can be modeled by a linear ODE as shown in Equation (8):

$$\frac{d\mathbf{y}}{dx} = \begin{bmatrix} -0.1 & -1 \\ 1 & -0.1 \end{bmatrix} \mathbf{y}, \tag{8}$$

in which, $x \in \mathbb{R}$ is the input and can be regarded as the order of data points, $\mathbf{y} \in \mathbb{R}^2$ is the two-dimensional output and can be regarded as the position of data points.

As we discussed in Section 2.1, data points following Equation (8) can be generated by Equation (9), in which $\boldsymbol{\epsilon} \in \mathbb{R}^2$ is the Gaussian random noise controlling the variations of generated data points;

$$\mathbf{y}_N = 4 \times \text{ODESolve}\left(\mathbf{y}_0, \frac{d\mathbf{y}}{dx}, x_0, x_N\right) + \boldsymbol{\epsilon}. \tag{9}$$

Our proposed NP-ODE is robust in modeling data points with various levels of variations. To demonstrate its robustness, the mean of $\boldsymbol{\epsilon}$ is set to zero and the standard deviation of $\boldsymbol{\epsilon}$ is set to $0.01,\ 0.02,$ and 0.1. For each noise level, 200 data points are generated for training and validating the method. The generated data points are visualized in Figure 6, in which the first subplot from the left shows the spiral without noise, the scatters in the other three subplots are data points sampled from this spiral with various noise levels. We can find out that when the standard deviation of noise equals 0.1, the pattern of the sampled data points is hard to identify.

4.2. Evaluation metrics

The Root Mean Square Error (RMSE) is selected to evaluate the predictive accuracy. The Confidence Interval (CI) within one standard deviation is chosen to conduct uncertainty quantification.

The expression of RMSE is given in Equation (10), in which $\mathbf{y}_i$ is the real position of the ith data point, $\bar{\mathbf{y}}_i$ is the predicted position of the ith data point, and $\|.\|_2^2$ is the square of the l_2 norm. A smaller value of RMSE on test points indicates that the model can predict more accurately:

$$\text{RMSE} = \sqrt{\frac{1}{2N}\sum_{i=1}^{N} \|\mathbf{y}_i - \bar{\mathbf{y}}_i\|_2^2}\ . \tag{10}$$

The CI within one standard deviation is selected for uncertainty quantification, which is denoted as $(\bar{\mathbf{y}}_i - \hat{\boldsymbol{\sigma}}_i,\ \bar{\mathbf{y}}_i + \hat{\boldsymbol{\sigma}}_i)$, and $\hat{\boldsymbol{\sigma}}_i \in \mathbb{R}^2$ is the predicted standard deviation in two dimensions. To show the robustness of the proposed NP-ODE, it should recover the underlying patterns (spiral without noise) from noisy data points. If the generated CI can cover the spiral curve, it indicates the corresponding method is robust to the noise.

4.3. Comparison with benchmark methods

To show the robustness of the proposed NP-ODE in modeling data points with different noise levels, we compare its performance on the simulated data points to the original NPs. In this section, we introduce the details of the experiment and discuss the results from the perspectives of mean prediction performance and UQ, respectively.

4.3.1. Experiment design

As shown in Figure 6, there are three noise levels of the simulated data with the standard deviation of noise equalling $0.01,\ 0.02,\ 0.1,$ respectively. For each noise level, 200

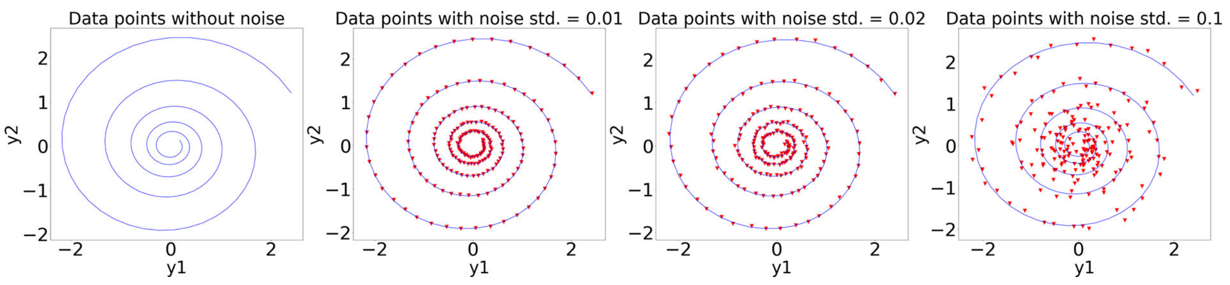


Figure 6. Visualization of generated data points.

Table 1. Results comparison (RMSE).

	Standard deviation of noise		
	0.01	*0.02*	*0.1*
NPs	0.0146	0.0268	0.2778
NP-ODE	**0.0136**	**0.0256**	**0.1171**

data points are generated, in which 150 data points are randomly selected to train the model, and the residual 50 data points test the model. To obtain a fair comparison, the details of experiments using NP-ODE and NPs remain the same. During the training phase, part of the training data are randomly selected as the context (observed) data points with both the input x_i and response $\mathbf{y}_i$ feeding into the model, and the rest of the training data are target (unobserved) data points with only input x_i feeding into the model. During the testing phase, all the training data serves as the context data points, and the testing data are target data points.

4.3.2. Mean prediction comparison

The mean prediction comparison on test data points between the proposed NP-ODE and the NPs are shown in Table 1. We can conclude that under different noise levels, the proposed NP-ODE can consistently generate more accurate predictions than the original NPs. Furthermore, when the standard deviation of noise increases, the prediction improvement is more significant.

4.3.3. UQ comparison

In Figure 7, we plot the generated curves on both training and testing data points along with the CI within one standard deviation. In Figure 7, the rectangles denote testing points and the stars denote training points; the curves are spiral without noise, which is the underlying pattern of noisy data points; the curves are generated by the NP-ODE or NPs on both training and testing data points; the cyan shadow represents the CI within one standard deviation in the dimension of y_1, and the yellow shadow represents the CI within one standard deviation in the dimension of y_2. The left and right columns of Figure 7 contain the results from the NP-ODE and the NPs, respectively. The first row indicates the training and testing data points (with noise) and the underlying spiral without noise. The middle two rows compare the NP-ODE and the NPs from the perspective of UQ in each dimension y_1 and y_2, respectively. The last row can be regarded as the combination of the middle two rows. Compared with the NPs, the proposed NP-ODE successfully captures the patterns of noisy data points, and the generated CI covers the underlying spiral curve. Note that Figure 7 only shows the modeling results on noisy data points with the noise standard deviation as 0.1. The results on other noise levels are listed in the supplementary Appendix III. Figure 7 shows that the generated trajectory from the proposed NP-ODE is consistent with the actual spiral curve. Via Table 1 and Figure 7, we can conclude that the proposed NP-ODE shows its strength in modeling data points governed by differential equations and demonstrates the robustness in revealing underlying patterns with noisy data points.

5. Case study

To further evaluate the strength and effectiveness of the proposed NP-ODE, a case study is conducted to build a surrogate model of FEA simulations and perform UQ. In this section, we introduced FEA simulations on material corrosion analysis for new materials design. The implementation details of NP-ODE are discussed, including the model structure and parameters analysis. Additionally, evaluation metrics are selected to measure model performance. Finally, the comparisons among our proposed model and benchmark models are discussed.

5.1. Introduction to FEA for tribocorrosion

Tribocorrosion is a material degradation process involving both mechanical wear and corrosion of the material, which jeopardizes a material's long-term sustainability and structural integrity. Tribocorrosion analysis is very important for design and manufacturing systems. The synergetic effects of mechanical damage and corrosion can cause more severe material degradation than the sum of pure wear and corrosion. To investigate the effects of a material's mechanical and electrochemical properties on their tribocorrosion behavior, an FEA model is developed to simulate both the dynamic process of wear and the time-dependent evolution of a corroding surface during tribocorrosion. We conduct an experimental study of two aluminum alloys (with 5 wt% Mn and 20 wt% Mn, respectively). The scheme of FEA and the meshed geometry are shown in Figure 8. The model first simulates a scratching wear process and produces results including the wear volume loss, and surface and subsurface stress and strain caused by the process. A phenomenological

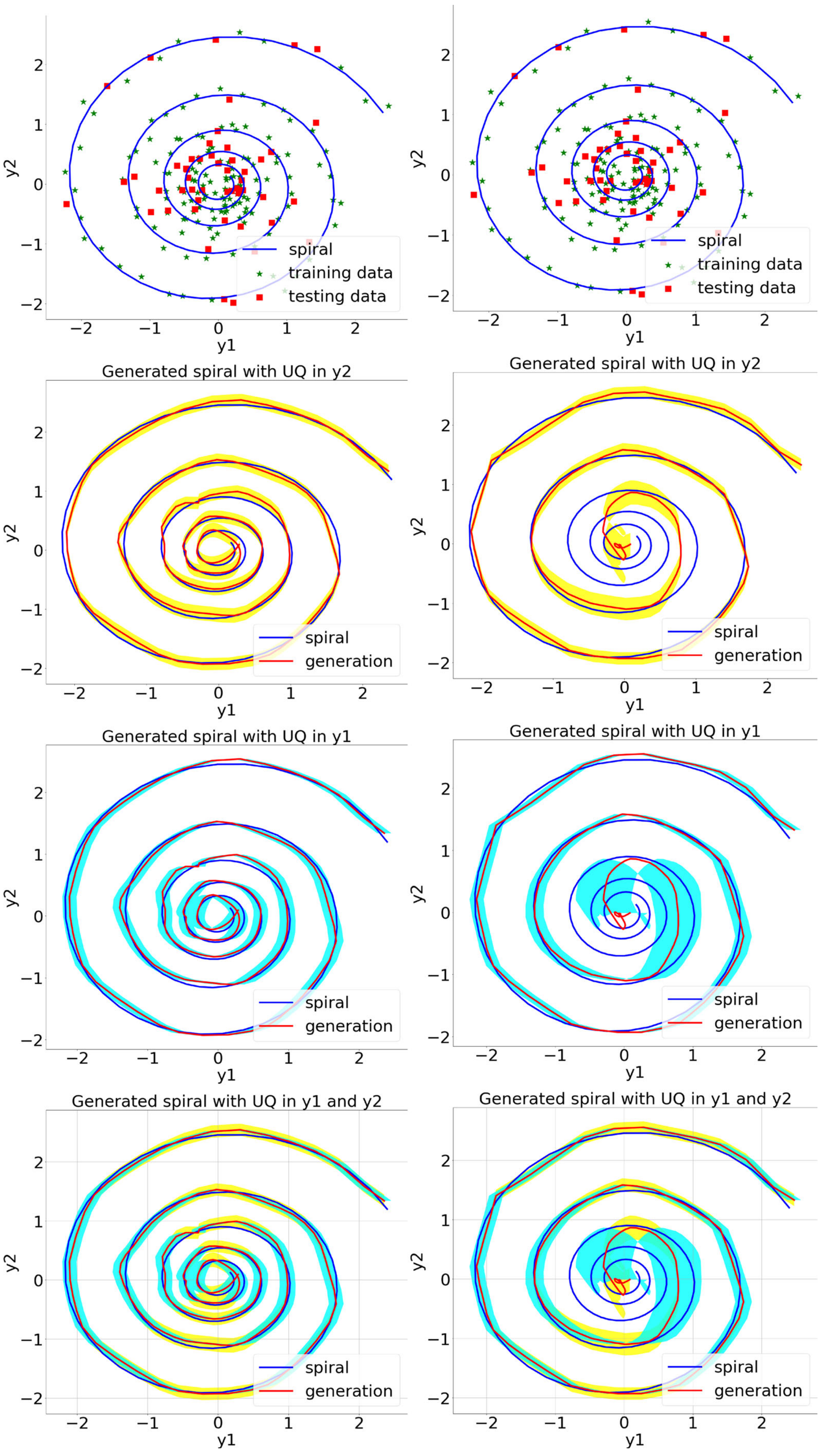


Figure 7. UQ comparison with noise $\sigma = 0.1$; **Left:** result from NP-ODE; **Right**: result from NPs.

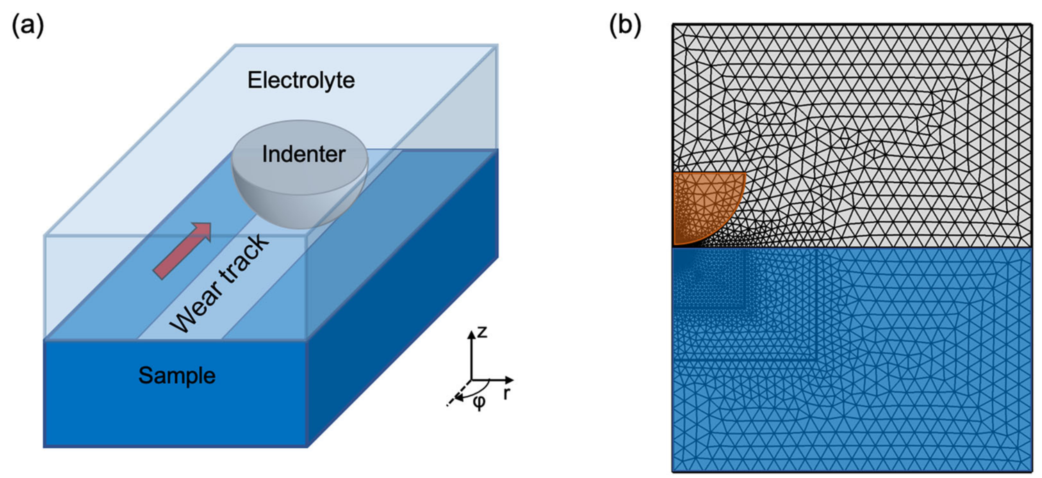


Figure 8. (a) Schematic diagram of FEA for tribocorrosion test; and (b) meshing setup of tribocorrosion test.

model reflecting the change in anodic potential caused by plastic strain is used to incorporate the wear–corrosion synergy. The corrosion process simulation considers the impact on the electrochemical state of the system caused by mechanical deformation.

5.2. Dataset introduction and preprocessing

To investigate the influence of a material's properties on tribocorrosion rate, six individual parameters were taken into consideration, including mechanical (Young's modulus and yield strength) and corrosion (anodic and cathodic Tafel slope and exchange current densities) parameters. The final output of the model is the material loss rate caused by tribocorrosion. In this case, the material loss caused by corrosion and wear–corrosion synergy is of interest. It can also be expressed as total material loss due to tribocorrosion minus that due to pure wear.

The effects of mechanical, anodic corrosion, and cathodic corrosion parameters were investigated separately. For the variation of mechanical parameters, Young's modulus was swept from 55 MPa to 95 MPa with a 5 MPa step size, the yield strength from 1.0 MPa to 5.0 MPa with a 0.5 MPa step size, while the corrosion parameters remained constant. For the variation of corrosion parameters, the cathodic Tafel slope was varied from -280 to -210 mV/decade, the cathodic exchange current density from 2.0×10^{-8} to 2.0×10^{-7} A/cm^2, the anode Tafel slope from 250 to 290 mV/decade, and the anodic exchange current density from 1.0×10^{-13} to 5.0×10^{-13} A/cm^2. Each combination of mechanical parameters and corrosion parameters will render a different output of material loss rate.

After obtaining the FEA dataset, data preprocessing is conducted to make it ready for surrogate modeling. In the dataset, there are 106 simulations. Each simulation contains six input variables representing the mechanical and corrosion properties and one output variable representing the corrosion rate. Since the input and output variables have different physical meanings and various significant scales, normalization is applied to the raw data to transform all the variables into $[-2, 2]$. The expression of normalization is shown in Equation (11):

$$x_{\text{norm}} = \left(\frac{x - x_{\min}}{x_{\max} - x_{\min}} \times 4 \right) - 2. \tag{11}$$

In Equation (11), x represents a single variable, $x_{\max}$ and $x_{\min}$ are the maximum and minimum value of x, respectively. After normalization, 20 simulations are randomly selected as the testing data and the rest 86 simulations are training data.

5.3. Parameter analysis and selection

The implementation details of the proposed NP-ODE will be introduced in this section. The number of parameters will be further analyzed and compared with original NPs given the specific model structure; the selection of parameters in the Euler's method for solving NP-ODE is also discussed.

Compared with the original NPs, the NP-ODE uses fewer parameters, due to the incorporation of the Neural-ODE as the decoder. In our experiment, the convolutional layer (Conv layer) is selected as the basic building block of the Neural-ODE to capture the derivative of continuous feature transformation. The reasons for this observation can be summarized into three aspects. First, the original NPs use the MLP as the decoder, in which the number of parameters is determined by the dimensions of the input and output feature vectors. In contrast, the number of parameters in the Conv layer is influenced by the size and the number of convolutional filters, which is not determined by the input and output. To this extent, especially for those high-dimensional features, the choice of the Conv layer can reduce the number of parameters. Second, the Neural-ODE uses a limited number of layers to model the derivative and mimic the continuous transformation, whereas the MLP in NPs discretizes the continuous transformation by the multi-layer structure. Thus, the number of parameters in NPs is determined by the model depth, whereas the depth in the Neural-ODE will not influence parameters. Third, when comparing the single Conv layer and FC layer (the building block of the

Table 2. Comparison of decoder parameters between NPs and NP-ODE.

Model	*Decoder Layer*	*Weight Matrix*	*# Parameters*
NPs	FC layer $\times$ 3	$(384, 384)$	147 456
		$(384, 384)$	147 456
		$(384, 384)$	147 456
NP-ODE	Conv layer 1	$(1, 128, 3, 1)$	384
	Conv layer 2	$(129, 128, 3, 1)$	49 536
	Conv layer 3	$(129, 128, 3, 1)$	49 536

MLP), the choice of the Conv layer may be hindered, as it only captures local features limited by the size of the convolutional filter. However, since the Neural-ODE can be regarded as the infinite-layer NN, the global features can be captured hierarchically, given the infinite number of Conv layer.

For example, in the decoder of NP-ODE, we use three Conv layers to build the ODE network to estimate the derivatives of feature transformation. By combining the ODE network with the ODE solver, the Neural-ODE can mimic the continuous feature transformation, whereas in the decoder of NPs, we use three FC layers to discretize the continuous feature transformation. Suppose the input feature of the decoder is of shape $(1, 384, 1)$, the detailed model structure, shape of the weight matrix of each layer, and the number of parameters of each layer are summarized in Table 2. From the comparison, we find that the number of parameters in the decoder of NPs is $147\ 456 \times 3 = 442\ 368$, whereas the number of parameters in the decoder of NP-ODE is $384 + 49\ 536 \times 2 = 99\ 456$. Moreover, the number of parameters in the decoder of NPs will further increase with the increase of model depth. In contrast, the model depth will not influence the number of parameters in the decoder of NP-ODE. Since the dataset generated from FEA simulations are often scarce, fewer parameters in the proposed NP-ODE can reduce the possibility of overfitting.

Euler's method is selected as the ODE solver in our experiment as shown in Equation (7). The D_0, D_N, ΔD are important parameters influencing the performance of Euler's method, they represent the input model depth, output model depth, and step size, respectively. Theoretically, the difference between the input and output depth $(D_N - D_0)$ along with the step size ΔD determine the accuracy of Euler's method. Given a specific value of $(D_N - D_0)$, a smaller step size will improve the accuracy and increase the training time. Considering the trade-off between accuracy and efficiency, we select the $(D_N - D_0)$ as 1 and ΔD as 0.05, which is a good combination based on the model's performances in prediction accuracy (sections 5.5.3, 5.5.4) and computational cost (section 5.5.5).

5.4. Evaluation metrics

Two evaluation metrics are selected and applied to the testing data. They are Mean Absolute Percentage Error (MAPE) for predictive accuracy and 95% CI for UQ.

The expression of MAPE is given as Equation (12). The MAPE is a relative error evaluating the relative difference between the predicted error and real value. It can be regarded as eliminating the differences among data scales and treating each data point equally. In Equation (12), N represents the number of testing data points, $\overline{y_i}$ is the predicted mean value of the ith data point:

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \overline{y_i}}{y_i}\right|. \qquad (12)$$

The output of NP-ODE is the distribution of predicted values which consists of predicted mean $\bar{y}$ and predicted standard deviation $\hat{\sigma}$. The CI can be generated for UQ. We select the 95% CI, which is calculated by $(\bar{y} - 1.96\hat{\sigma},\ \bar{y} + 1.96\hat{\sigma})$ for each testing data point. There are three criteria to evaluate the quality of the CI: (i) the real value of the testing data point lies in the CI of the predicted value; (ii) the real value is close to the center of the CI (predicted mean $\bar{y}$); and (iii) a smaller $\hat{\sigma}$ ensuring the coverage of the real value of FEA simulations.

5.5. Comparison with benchmark methods

We apply the proposed NP-ODE and other benchmark methods to build surrogates of the FEA simulations and compare their performance. The original NPs, GP with Matern kernel, and GP with Polynomial kernel are selected as benchmark methods. In this section, we will give a brief introduction of these benchmark methods, illustrate the design of the experiment, and compare their performances.

5.5.1. Introduction to benchmark methods

The first benchmark method is the original NPs and it is selected to show the advantages of incorporating Neural-ODE as the decoder. The other two benchmark methods belong to the family of the GP. In the GP, instead of finding a deterministic function, it derives the probability distribution over all possible functions that fit the data. The basic steps are: (i) to specify a prior distribution on the function space; (ii) based on the Bayesian rule, calculate the posterior distribution using training data; (iii) use the posterior distribution to make an inference on testing data. The choice of covariance function is a significant factor influencing the performance of the GP. We selected GPs with Matern and Polynomial kernels as the second and third benchmark methods. These two kernel functions are commonly used to capture nonlinear relationships.

5.5.2. Experiment design

In the experiment, each FEA simulation can be formulated as a data point $(\mathbf{x},\ y)$, $\mathbf{x} \in \mathbb{R}^6$, $y \in \mathbb{R}$. Among all the 106 simulations, 20 simulations are randomly selected as the testing data, and the remaining 86 simulations are training data. To test the model's ability to explore limited data points, we repeat the experiment, change the number of training points available to models, and evaluate the performance on the same testing data to check the robustness of models. The number of training samples is iteratively increased. Thirty samples out of 86 training data are randomly selected at first, and then the sample size is increased to 50, 60, 70, and 80 by successively randomly adding new data points from the training data. The comparison of results among all the methods is introduced next.

5.5.3. Mean predictions comparison

The outputs of our proposed NP-ODE and benchmark methods are all distributions over a predicted value. To numerically evaluate the prediction results, we calculate the MAPE between the predicted mean $\bar{y}$ and real value y on testing data. The results are summarized in Table 3.

Table 3. Results comparison (MAPE %).

# Train	*NP-ODE*	*NPs*	*GP (Matern)*	*GP (Poly)*
30	**5.5229**	9.2862	11.8461	19.3152
50	**2.7286**	4.9690	7.1928	10.3434
60	**2.5886**	4.8660	6.6770	14.1264
70	**2.2121**	3.2018	5.1234	2.3593
80	**2.1235**	3.2364	4.4351	2.4418

In Table 3, the first column represents the number of training samples available to models, and each row is the testing MAPE for the proposed NP-ODE and benchmarks. We can conclude that: (i) given the same training samples, our proposed NP-ODE outperforms the benchmark methods consistently; (ii) with more training samples, all the methods tend to have a better performance. In addition, the performance of NP-ODE has the smallest variation and the best accuracy with the different number of training samples. Our proposed model can explore the features in training samples better and has a more robust performance with the different

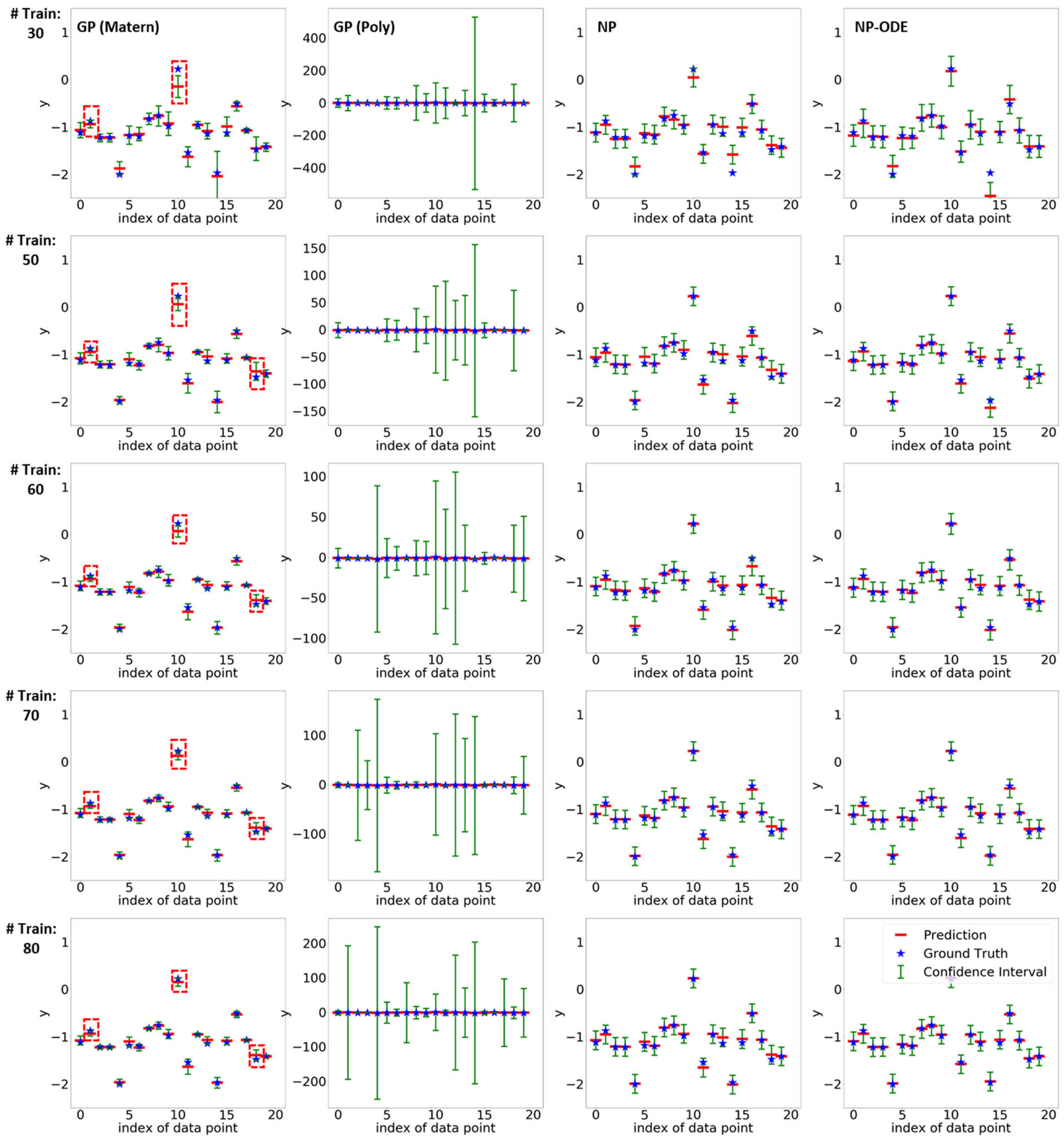


Figure 9. Visualization of UQ.

number of training samples. Given the predicted mean and standard deviation on each testing data point, the 95% CI is generated for uncertainty quantification. The visualization of UQ is shown in Figure 9, in which the star represents the ground-truth value, the line in the middle represents the predicted mean, and the capped line represents the 95% CI ($\pm 1.96\hat{\sigma}$). From Figure 9, we can find that: (i) with more training data, the predicted mean value (line in the middle) tends to get closer to the real value (star); (ii) the GP with the Matern kernel tends to underestimate the standard deviation, and some of the real values lie out of the boundary of the 95% CI (highlighted in dashed boxes); (iii) the GP with the Polynomial kernel tends to overestimate the standard deviation, which makes the CIs make no sense in applications; (iv) the NPs and NP-ODE generally generate a reasonable standard deviation and predicted mean at each data point, which makes the real value close to the predicted mean and the CI gives a reasonable estimation of the uncertainty.

5.5.4. UQ comparison

In summary, the NP-ODE keeps the real value close to the center of the CI and generates a reasonable estimation of uncertainty. The NP-ODE gives the most accurate prediction on testing data points. When feeding 80 training samples into the model, the predicted 95% CI has the best coverage on testing data points.

5.5.5. Computational cost

The training time of the proposed NP-ODE is mainly determined by the number of training iterations. Iteration here denotes a complete forward and backward propagation to update the model parameters. The experiments are conducted on a single NVIDIA TITAN V GPU, and it takes around 0.1 seconds to run a single iteration in training NP-ODE. The training process commonly has 10 000 iterations, so that it takes around 17 minutes to train the NP-ODE. For comparison, the training process of the original NPs takes around 5 minutes for 10 000 iterations, and the training process of the GP takes around 0.05 seconds.

It should be noted that the training process for surrogate models is the most time-consuming step and is often taken offline. As long as the model is ready to use, there is no need to repeat the training process. The computational cost of the NP-ODE in predicting 20 testing data points is 0.2 seconds, which is comparable to the original NPs and the GP. More importantly, comparing the time efficiency of surrogate models to the FEA method is more practically meaningful. Compared with the FEA method, the trained NP-ODE significantly reduces the computational cost of predicting results on new inputs from hours to seconds.

6. Conclusion

Despite the strength and accuracy of the FEA method, its applications are hindered by the high computational cost and lack of ability in UQ. Since uncertainties inevitably exist in engineering systems, UQ is essential in system modeling. The Monte Carlo method is a standard approach to conduct UQ. However, the Monte Carlo method suffers from significantly growing computational time when repeating simulations. The existing surrogates built for UQ of FEA are mainly based on the GP and its variants. Although it reduces computational cost compared with the FEA method, it suffers from a lack of interpretability and is unable to handle large volume and high-dimensional data.

This article proposes a physics-informed stochastic surrogate NP-ODE to model the FEA simulations as well as evaluating the uncertainties of the output. In the NP-ODE, the basic structure enables the model to generate distributions of the output for UQ, and the Neural-ODE is incorporated as the decoder. It improves the model's ability to solve systems governed by differential equations. In the case study, we select MAPE and 95% CI to evaluate the predictive accuracy and UQ. Based on the case study, the advantages of our proposed NP-ODE can be summarized into several aspects: (i) compared with the GP and its variants, our proposed NP-ODE shows a better ability in exploring limited training samples and has a robust performance in both predictive error and UQ when decreasing the training samples; (ii) compared with the original NPs, the incorporation of Neural-ODE reduces the number of model parameters and enables our proposed NP-ODE to better model FEA simulations; (iii) the proposed NP-ODE solves differential equations in its decoders, so it is more physically close to the mechanism of the original FEA.

There are several limitations of the proposed NP-ODE. First, incorporating Neural-ODE might introduce extra computational cost if we select a small value of step size ΔD. In our experiment, the influence is mitigated by tuning the value of ΔD to find a balance between accuracy and efficiency. Second, given the accurate results of the simulation study and case study, the assumption of Euler's method is to fix the value of ΔD in solving NP-ODE. Other numerical methods with an adaptive step size can be the alternatives to solve NP-ODE. For future extensions, the proposed NP-ODE may provide a prototype about incorporating UQ and stochastic surrogates for FEA simulations. Following this prototype, it will be interesting to design stochastic surrogates for systems governed by PDEs.

Data and code availability

The dataset and codes for this paper are available in the supplementary files (doi.org/10.6084/m9.figshare.13828514). They are also available in Github https://doi.org/10.6084/m9.figshare.13828514.

Funding

This research was supported by the US National Science Foundation under Grant CMMI-1855651. This research was also partially supported by the US National Science Foundation under CMMI-2035038. The computational resource used in this work is provided by the advanced research computing at Virginia Polytechnic Institute and State University.

Notes on contributors

Yinan Wang received a BS degree in electrical engineering and automation from Xi'an Jiaotong University, Xi'an, China, in 2017, an MS in electrical engineering from Columbia University, New York, NY, USA, in 2019. Currently, he is a PhD student at the Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA, USA. His research interests include data analytics, system pattern recognition, machine learning techniques in material/system design, and advanced manufacturing. Mr Wang is a member of INFORMS, IISE, and ASA.

Kaiwen Wang received a BS degree in applied physics from the University of Science and Technology of China, in 2017. Currently, he is a PhD student at the Material Science and Engineering Department of Virginia Tech, Blacksburg, VA. His research interests include multiscale and multiphysics simulations on tribocorrosion behavior of metals, alloys, and structural materials, specifically using finite element method and molecular dynamics.

Dr. Wenjun Cai is an assistant professor in the Department of Materials Science and Engineering at Virginia Tech, USA. She received her BS in materials science from Fudan University in 2005 and her PhD in materials science engineering from the University of Illinois at Urbana-Champaign in 2010. Her research interests are in physical metallurgy; corrosion and tribocorrosion; materials deformation and degradation under extreme conditions; materials characterization; mechanical testing and fracture mechanics; and thin films and coatings. She is the recipient of the 2017 TMS Young Leaders Professional Development Award, 2016 Outstanding Faculty Award of the University of South Florida, and 2015 CAREER Award of National Science Foundation.

Dr. Xiaowei Yue received a BS degree in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2011, an MS in power engineering and thermophysics from the Tsinghua University, Beijing, China, in 2013, an MS in statistics, PhD in industrial engineering with a minor machine learning from the Georgia Institute of Technology, Atlanta, USA, in 2016 and 2018. Currently, he is an assistant professor at the Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, USA. His research interests are focused on engineering-driven data analytics for advanced manufacturing. He is a recipient of *IEEE Transactions on Automation Science and Engineering* Best Paper Award and several other best paper awards. He is also a recipient of Mary G. and Joseph Natrella Scholarship from ASA, and FTC Early Career Award from ASQ. He serves as an associate editor for the *Journal of Intelligent Manufacturing* and the *IISE Transactions.*

ORCID

Yinan Wang http://orcid.org/0000-0002-4079-1658
Kaiwen Wang http://orcid.org/0000-0003-4765-8726
Wenjun Cai http://orcid.org/0000-0002-9457-8705
Xiaowei Yue http://orcid.org/0000-0001-6019-0940

References

Ankenman, B., Nelson, B.L. and Staum, J. (2010) Stochastic kriging for simulation metamodeling. *Operations Research*, **58**(2), 371–382.

Bahdanau, D., Cho, K.H. and Bengio, Y. (2015) Neural machine translation by jointly learning to align and translate, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015) Weight uncertainty in neural network, in *Proceedings of the 32nd International Conference on Machine Learning* (*ICML*), pp. 1613–1622, PMLR. http://proceedings.mlr.press/v37/blundell15.html.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J. and Duvenaud, D. (2018) Neural ordinary differential equations, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (*NeurIPS*), pp. 6572–6583, Curran Associates Inc., Red Hook, NY.

Dong, J., Qin, Q.H. and Xiao, Y. (2020) Nelder–Mead optimization of elastic metamaterials via machine-learning-aided surrogate modelling. *International Journal of Applied Mechanics*, **12**(1), 2050011.

Gal, Y. and Ghahramani, Z. (2016) Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in *Proceedings of the 33rd International Conference on Machine Learning* (*ICML*), pp. 1050–1059, PMLR. http://proceedings.mlr.press/v48/gal16.html

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y.W., Rezende, D. and Eslami S.M.A. (2018) Conditional neural processes, in *Proceedings of the 35th International Conference on Machine Learning* (*ICML*), pp. 1704–1713, PMLR. http://proceedings.mlr.press/v80/garnelo18a.html

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D.J., Eslami, S.M.A. and Teh, Y.W. (2018) Neural processes. *arXiv preprint arXiv:1807.01622*.

Griffiths, D.F. and Higham, D.J. (2010) *Numerical Methods for Ordinary Differential Equations*: *Initial Value Problems*, Springer, London, UK.

Hoang, T.N., Hoang, Q.M. and Low, B.K.H. (2015) A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data, in *Proceedings of the 32nd International Conference on Machine Learning* (*ICML*), pp. 569–578, PMLR. http://proceedings.mlr.press/v37/hoang15.html

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O. and Teh, Y.W. (2019) Attentive neural processes. *arXiv preprint arXiv:1901.05761*.

Liang, L., Liu, M., Martin, C. and Sun, W. (2018) A deep learning approach to estimate stress distribution: A fast and accurate surrogate of finite-element analysis. *Journal of the Royal Society Interface*, **15**(138), 20170844.

Liu, H., Ong, Y.S., Shen, X. and Cai, J. (2020) When Gaussian process meets big data: A review of scalable GPs. *IEEE Transactions on Neural Networks and Learning Systems*, **31**(11), 4405–4423.

Loose, J.P., Chen, N. and Zhou, S. (2009) Surrogate modeling of dimensional variation propagation in multistage assembly processes. *IIE Transactions*, **41**(10), 893–904.

Mahadevan, S. and Liang, B. (2011) Error and uncertainty quantification and sensitivity analysis in mechanics computational models. *International Journal for Uncertainty Quantification*, **1**(2), 147–161.

Rubinstein, R.Y. and Kroese, D.P. (2017) *Simulation and the Monte Carlo Method*, Wiley, Hoboken, NJ.

Snelson, E. and Ghahramani, Z. (2006) Sparse Gaussian processes using pseudo-inputs, in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pp. 1257–1264, Curran Associates Inc., Red Hook, NY.

Su, G., Peng, L. and Hu, L. (2017) A Gaussian process-based dynamic surrogate model for complex engineering structural reliability analysis. *Structural Safety*, **68**, 97–109.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) Attention is all you need, in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (*NeurIPS*), pp. 6000–6010, Curran Associates Inc., Red Hook, NY.

Wang, G.G. and Shan, S. (2006) Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, **129**(4), 370–380.

Wang, H., Yuan, J. and Ng, S.H. (2019) Gaussian process based optimization algorithms with input uncertainty. *IISE Transactions*, **52**(4), 377–393.

Wang, K., Wang, Y., Yue, X. and Cai, W. (2021) Multiphysics modeling and uncertainty quantification of tribocorrosion in aluminum alloys. *Corrosion Science*, **178**, 109095.

Wang, L., Chen, X., Kang, S., Deng, X. and Jin, R. (2020) Meta-modeling of high-fidelity FEA simulation for efficient product and process design in additive manufacturing. *Additive Manufacturing*, **35**, 101211.

Wang, W., Yue, X., Haaland, B. and Wu, C.J. (2020) Gaussian process with input location error and applications to composite fuselage shape control. *arXiv preprint arXiv:2002.01526*.

Yildiz, C., Heinonen, M. and Lahdesmaki, H. (2019) ODE2VAE: Deep generative second order ODEs with Bayesian neural networks, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (*NeurIPS*), Curran Associates Inc., Red Hook, NY.

Yue, X. and Shi, J. (2018) Surrogate model–based optimal feed-forward control for dimensional-variation reduction in composite parts assembly processes. *Journal of Quality Technology*, **50**(3), 279–289.

Yue, X., Wen, Y., Hunt, J.H. and Shi, J. (2018) Surrogate model-based control considering uncertainties for composite fuselage assembly. *Journal of Manufacturing Science and Engineering*, **140**(4), 041017.

Yue, X., Wen, Y., Hunt, J.H. and Shi, J. (2020) Active learning for Gaussian process considering uncertainties with application to shape control of composite fuselage. *IEEE Transactions on Automation Science and Engineering*, **18**(1), 36–46.

Zienkiewicz, O.C., Taylor, R.L. and Zhu, J. (2013) *The Finite Element Method: Its Basis and Fundamentals*, seventh edition, Elsevier Butterworth-Heinemann, Burlington, MA.