Final Paper to *ASME Journal of Computing and Information Science in Engineering*

Surrogate Modeling of Nonlinear Dynamic Systems: A Comparative Study

Ying Zhao[1], Chen Jiang[1], Manuel A. Vega[2, 3], Michael D. Todd[2], and Zhen Hu[1*]

[1]Department of Industrial and Manufacturing Systems Engineering, University of Michigan-Dearborn, Dearborn, MI 48128, USA
[2]Department of Structural Engineering, University of California San Diego, La Jolla, CA 92093, USA
[3]Advanced Engineering Analysis Group, Los Alamos National Laboratory, Los Alamos, NM 87545

* Corresponding author: 2340 HPEC, University of Michigan-Dearborn, Dearborn, MI 48128, USA, Tel:+1-313-583-6312, Email: zhennhu@umich.edu

AUTHORS INFORMATION:

Ying Zhao
Ph.D. Student
Department of Industrial and Manufacturing Systems Engineering,
University of Michigan-Dearborn,
2340 Heinz Prechter Engineering Complex,
Dearborn, MI, 48128
E-mail: yinzhao@umich.edu

Chen Jiang
Postdoctoral Research Scholar
Department of Industrial and Manufacturing Systems Engineering,
University of Michigan-Dearborn,
2340 Heinz Prechter Engineering Complex,
Dearborn, MI, 48128
E-mail: chejiang@umich.edu

Manuel A. Vega
R&D Engineer
Advanced Engineering Analysis Group,
Los Alamos National Laboratory,
Los Alamos, NM 87545
E-mail: mvegaloo@lanl.gov

Michael D. Todd
Professor
Department of Structural Engineering
University of California San Diego
La Jolla, CA 92093, USA
E-mail: mdtodd@eng.ucsd.edu

Zhen Hu
Corresponding Author
Assistant Professor
Department of Industrial and Manufacturing Systems Engineering,
University of Michigan-Dearborn,
2340 Heinz Prechter Engineering Complex,
Dearborn, MI, 48128
E-mail: zhennhu@umich.edu

Abstract

Surrogate models play a vital role in overcoming the computational challenge in designing and analyzing nonlinear dynamic systems, especially in the presence of uncertainty. This paper presents a comparative study of different surrogate modeling techniques for nonlinear dynamic systems. Four surrogate modeling methods, namely Gaussian process (GP) regression, a long short-term memory (LSTM) network, a convolutional neural network (CNN) with LSTM (CNN-LSTM), and a CNN with bidirectional LSTM (CNN-BLSTM), are studied and compared. All these model types can predict future behavior of dynamic systems over long periods based on training data from relatively short periods. The multi-dimensional inputs of surrogate models are organized in a nonlinear autoregressive exogenous model (NARX) scheme to enable recursive prediction over long periods, where current predictions replace inputs from the previous time window. Three numerical examples, including one mathematical example and two nonlinear engineering analysis models, are used to compare the performance of the four surrogate modeling techniques. The results show that the GP-NARX surrogate model tends to have more stable performance than the other three deep learning-based methods for the three particular examples studied. The tuning effort of GP-NARX is also much lower than its deep learning-based counterparts.

Keywords: Surrogate modeling; Dynamic system; Data-driven; Time series

1. INTRODUCTION

Modeling of dynamic systems plays a vital role in the analysis, design, health monitoring, and control of various dynamic engineering systems, such as chemical processes [1], civil infrastructure [2], battery systems [3], and vibratory mechanical systems [4]. A dynamic system can be linear or nonlinear. The modeling of nonlinear dynamic systems, in general, is more complicated than that of linear systems.

In order to describe the complex nonlinear behaviors of dynamic systems, various approaches have been developed for different types of systems using analytical methods or sophisticated computer simulations. Analytical models obtained through theoretical modeling based on simplification and assumptions, usually have a low requirement on the computational effort. However, the prediction accuracy is usually sacrificed due to model simplifications [5]. With the development of high-performance computing and advanced computational methods, high-fidelity computer simulations are becoming more common in the design and analysis of various dynamic systems. They play an essential role in the design of reliable complex nonlinear engineering systems, such as hypersonic aircraft [6], off-road vehicles [7], and large civil infrastructure [8]. While the sophisticated computer simulation models significantly increase the prediction accuracy, the required computational effort is also drastically increased, which poses challenges to the associated analysis, design, model updating (e.g., a digital twin), and model predictive control [9].

Aims to overcome the computational challenges introduced by the sophisticated computer simulation models, surrogate models constructed using machine learning (ML) techniques are usually used to substitute the original computer simulation models or experiments. As data-driven techniques, surrogate models construct a model of the original model by establishing a relationship between inputs and outputs of the complex dynamic system based on data. It provides a promising way of efficiently predicting the nonlinear dynamic behavior of various systems without sacrificing accuracy. It also allows for forecasting the nonlinear dynamic behavior in a reasonable time horizon in the future based on a certain amount of historical data. Due to these advantages of dynamic surrogate models, their applications can be seen in not only engineering design, but also many other fields where dynamic predictive models play an essential role, including disease transmission modeling [10], medical and environmental science [11], among others.

In the past decades, various surrogate modeling techniques have been developed to model nonlinear dynamic behavior. According to the fundamental differences in the modeling forms, the current approaches can be classified into two groups, namely *input-output* models and *state-space* models [12]. For surrogate models in input-output forms, they are constructed by directly describing the relationship between the inputs and outputs based on observation data [12]. For state-space models, the surrogate models are represented in state-space forms [12]. The state-space model-based methods can be further classified into two categories: linear state-space models and nonlinear state-space models. Various mature techniques have been developed for the learning of linear state-space models in the field of dynamic system identification [13]. For the learning of nonlinear state-space models, it usually requires a nonlinear transformation of a vector of past input/output samples to a state vector [14]. Due to the importance of state-space models in control, numerous approaches have been developed in recent years using machine learning and/or optimization-based methods to learn nonlinear state-space models. For instance, Masti and Bemporad developed an autoencoder-based approach for the learning of nonlinear state-space models [15]; Deshmukh and Allison proposed a derivative function-based surrogate modeling method to learn the nonlinear state-space models [16]; Gedon et al. proposed a deep state space model method for learning of nonlinear temporal models in the state-space form [17] , and an optimization-based approach is suggested in [18] for system identification of nonlinear state-space models. Both the input-output form and state-space form of surrogate models are widely used in the analysis and modeling of nonlinear dynamic systems. They have their own advantages and disadvantages. The input-output form of surrogate models does not require an explicit definition of a Markovian state or any information about the internal states. It is applicable to any type of dynamic systems with different complexities since it directly builds a function for the inputs and outputs. The disadvantage of the input-output form of surrogate models is that the order of the model is high due to the number of lags required to capture the relationship between inputs and outputs [18]. The nonlinear state-space form of surrogate models usually has a lower order (i.e. lower number of inputs) and is more efficient than its input-output counterparts. But it typically requires the definition of internal states and the surrogate modeling methods are much more complicated to implement in practice [18].

This paper focuses on surrogate models of nonlinear dynamic systems in the *input-output* form. In the past decades, a large group of methods have been developed for the modeling of dynamic systems in this form. For instance, autoregressive integrated moving average (ARIMA) [19] has gained considerable attention due to its predictive capability for time series models. ARIMA is derived from the autoregressive models (AR) [20], moving average models (MA) [21], and the autoregressive moving average models (ARMA) [22]. In general, the main limitation of these model classes is their imposed linear regression on past history and their deficiency in accounting for other input variables other than time. The nonlinear autoregressive model with exogenous input (NARX) models, which can overcome these limitations, have been proposed in the dynamics field based on artificial neural networks (ANN) [23] and Gaussian process regression (GP) [24]. For example, one approach is the surrogate model with nonlinear function approximation, like ANN, support vector regression [25], kernel support vector machine [26], and the other one is tree-based ensemble learning algorithm [27], such as decision tree [28] and random forest [28, 29]. In recent years, ANN with the robust nonlinear function such as multi-layer perceptron (MLP) network [30] has been extensively used in nonlinear dynamic predictive models for various applications. However, the lack of dependencies between inputs in the sequence processing affects the accuracy in long-term sequence prediction tasks. In contrast, GP models conquer the problems mentioned above and provide more accurate predictions in the NARX scheme. In the new era of deep learning (DL), various approaches have been proposed recently to construct surrogate models of nonlinear dynamic systems. DL surrogate models include, but are not limited to, recurrent neural network (RNN) [31], long short-term memory (LSTM) which is a type of RNN [32], convolutional neural network (CNN) [33], and hybrid structures that combines different deep neural networks [34].

Even though the methods as mentioned earlier have shown good performance in different applications, there is no generic surrogate modeling method that is applicable to all nonlinear dynamic systems and across different domains. Selecting an appropriate surrogate modeling method for a specific nonlinear dynamic system remains an issue that needs to be addressed. With a focus on data-driven approaches, this paper performs a comprehensive comparative study of different surrogate modeling methods under the NARX scheme to investigate the predictive capability of different methods. Four widely used approaches, namely GP-NARX, LSTM, CNN-LSTM, and CNN with bidirectional LSTM (CNN-BLSTM), are extensively compared using three numerical examples to investigate the advantages and disadvantages of different approaches. The three examples include a simple mathematical model, a duffing oscillator model, and a Bouc-Wen model. It is expected that the finding from this research will provide guidance and reference for the selection of surrogate models to reduce the computational effort in building predictive models for the design and analysis of nonlinear dynamic systems.

The remainder of this paper is organized as follows. Section 2 presents a generalized description of nonlinear dynamic models. Section 3 introduces surrogate models studied in this paper. Section 4 presents the comparative studies of different surrogate models. Finally, Section 5 summarizes the results and draws conclusions.

2. BACKGROUND

This section first presents the generalized dynamic predictive models in the NARX form. Then, various forecasting strategies, including one-step or multi-step methods, are briefly discussed.

2.1 Nonlinear Dynamic Predictive Model

A generalized nonlinear dynamic predictive model is defined as

$$y_i = f\left(\mathbf{x}_i, \boldsymbol{\theta}\right) + \nu_i, \tag{1}$$

where y_i is the output at time step t_i, $f(\cdot)$ is a nonlinear mapping from the regression vector $\mathbf{x}_i$ to the output space, $\boldsymbol{\theta}$ is an *s*-dimensional parameter vector included in $f(\cdot)$, the noise ν_i is used to account for the fact that the output y_i will not in practice be a deterministic function of past data. The regression vector $\mathbf{x}_i$ may be rewritten as [24]

$$\mathbf{x}_i = \varphi\left(u_i, u_{i-1}, \cdots, u_{i-q}, y_{i-1}, y_{i-2}, \cdots, y_{i-p}\right), \tag{2}$$

where $y_{i-j}, j = 1, 2, \cdots, p$ are the delayed samples of the measured output signal at time steps $t_{i-1}, t_{i-2}, \cdots, t_{i-p}$, $u_{i-k}, k = 1, 2, \cdots, q$ are the delayed samples of the measured input signal at time steps $t_{i-1}, t_{i-2}, \cdots, t_{i-q}$, u_i is the measured input value at the current time step, $\varphi(\cdot)$ is a nonlinear mapping from the measured input and output values to the regression vector, and p and q are the number of lags in the inputs and outputs, respectively.

According to the choice of regressors, current nonlinear dynamic predictive models in Eq. (1) can be classified into six main categories: (1) nonlinear finite impulse response (NFIR) models, which use only delayed input values u_{i-k} as regressors and are always stable due to the deterministic input values [35]; (2) NARX models, which use both delayed output signals y_{i-j} and input signals u_{i-k} as regressors and are also called series-parallel models [36]; (3) nonlinear output error (NOE) models, which use the estimation $\hat{y}_{i-j}$ of output value y_{i-j} and input signals u_{i-k} as regressors [37]; (4) nonlinear autoregressive and moving average model with exogenous input (NARMAX) models, which use y_{i-j} , u_{i-k} , and prediction error as regressors [38]; (5) nonlinear Box-Jenkins (NBJ) models [39]; and (6) nonlinear state-space models [18].

2.2 Predictive Model Forecast Strategies

In order to obtain accurate prediction of future of nonlinear dynamic behavior, various strategies have been developed to build dynamic predictive models. According to the number of future prediction time step(s), the current strategies can be classified into two groups: one-step approach and multi-step approach. The one-step approach predicts only the next time step of a time series, while the multi-step approach may predict any number of forward time steps. In general, the multi-step prediction approach is more likely to be adopted due to its capability of providing longer-term predictions, especially for reliability analysis/prognostics purposes.

The traditional direct multi-step forecasting is usually replaced by recursive strategies for multi-step forecasting to improve the prediction accuracy over a long-time horizon. The forecasting models with recursive strategy, in general, are better than the direct multi-step forecasting in terms of prediction accuracy and stability [40]. However, to overcome the limitation of error accumulation, some new techniques have been developed by integrating different types of dynamic models, such as multiple output strategies for multi-step forecasting [41] and the direct-recursive hybrid methods [42]. In what follows, we briefly review four classical multi-step time series forecasting methods.

2.2.1 *Direct multi-step forecast strategy*

The direct multi-step strategy considers a single-output model as follows [43]

$$\hat{y}_i = f_h(\mathbf{u}_i, y_{i-1}, y_{i-2}, \cdots, y_{i-p}, \boldsymbol{\theta}), \tag{3}$$

where $\mathbf{u}_i = [u_i, u_{i-1}, \cdots, u_{i-q}]$, $\hat{y}_i$ is the predicted output at time step t_i, $f_h(\cdot)$, $h = 1, 2, \cdots, H$ is the *h*-th single-output model from a total of H number of models, y_{i-j}, $j = 1, 2, \cdots, p$ are the measured output signals at time steps t_{i-j}, $j = 1, 2, \cdots, p$, and p is the number of time lags, as mentioned above. Through Eq. (3), predictions of maximally H time steps are obtained by combining predictions from each single-output model. For example, to obtain the prediction of the following two steps t_{i+1} and t_{i+2}, we may develop one model for forecasting at t_{i+1} and a separate model for forecasting at t_{i+2} as

$$\begin{aligned} \hat{y}_{i+1} &= f_1(\mathbf{u}_{i+1}, y_i, y_{i-1}, \cdots, y_{i-p+1}, \boldsymbol{\theta}), \\ \hat{y}_{i+2} &= f_2(\mathbf{u}_{i+2}, y_{i+1}, y_i, \cdots, y_{i-p+2}, \boldsymbol{\theta}). \end{aligned} \tag{4}$$

The drawback of the direct multi-step forecast strategy in Eq. (3) is that the H independent models result in predictions $\hat{y}_{i+1}, \hat{y}_{i+2}, \cdots, \hat{y}_{i+H}$ without statistical dependencies. It is more complicated than the recursive strategy and requires more computational time because of the required number of predictive models.

2.2.2 *Recursive multi-step forecast strategy*

The recursive multi-step forecast strategy learns a one-step model multiple times, where the prediction of the previous time step will be considered as one of the inputs at the next time step as follows [43]

$$\begin{aligned}\hat{y}_{i+1} &= f(\mathbf{u}_{i+1}, y_i, y_{i-1}, ..., y_{i-p+1}, \boldsymbol{\theta}),\\ \hat{y}_{i+2} &= f(\mathbf{u}_{i+2}, \hat{y}_{i+1}, y_i, y_{i-1}, ..., y_{i-p+2}, \boldsymbol{\theta}),\end{aligned} \tag{5}$$

where $y_{i-j},\ j = 0, 1, \cdots, p-1$ are the samples of the measured output signal at time step $t_{i-j},\ j = 0, 1, \cdots, p-1$. In Eq. (5), in order to predict the next two time steps t_{i+1} and t_{i+2}, the recursive multi-step strategy develops a one-step forecasting model $f(\cdot)$ at time step t_{i+1} and then uses it iteratively by adding the $\hat{y}_{i+1}$ into the input vector for the next forecast at t_{i+2}.

The recursive multi-step forecast strategy eliminates limitations of the direct multi-step forecast method. However, the estimation error will accumulate over time since the predicted values are used as inputs instead of the actual ones. The error accumulation would generally degrade the accuracy and amplify the uncertainty with the size of the prediction time horizon.

2.2.3 *Direct-recursive hybrid strategies*

The direct-recursive hybrid strategies combine the underlying principles of direct and recursive forecasting strategies. When this strategy is employed, the high-dimensional data may be considered as input variables. With the proper optimization algorithm, redundant information can be deleted to avoid overfitting problems. Based on both direct and recursive strategies, it can diminish dynamic system loss [29]. This strategy expands the input variables by replacing the true values with predicted ones using the recursive strategy, and it generates separate models with distinctive time horizons similar to the direct strategy. Thus, this method overcomes the shortcomings of the previous two strategies by avoiding error accumulation and maintaining the dependency of each estimated output. It is worth mentioning that this technique requires higher fidelity analysis and complex formulations because it demands the embedding size to be different for the whole range of prediction horizons. The hybrid strategy can be described as follows [43]

$$\begin{aligned}\hat{y}_{i+1} &= f_1(\mathbf{u}_{i+1}, y_i, y_{i-1}, ..., y_{i-p+1}, \boldsymbol{\theta}),\\ \hat{y}_{i+2} &= f_2(\mathbf{u}_{i+2}, \hat{y}_{i+1}, y_i, ..., y_{i-p+2}, \boldsymbol{\theta}),\end{aligned} \tag{6}$$

where y_{i-j}, $j = 0, 1, \cdots, p-1$ are samples of the measured output signal at a time step t_{i-j}, $\hat{y}_{i+1}$ and $\hat{y}_{i+2}$ are the predictions at time step t_{i+1} and t_{i+2}, $f_1(\cdot)$ and $f_2(\cdot)$ are separate predictive models.

2.2.4 *Multiple output strategy*

When the prediction of a time horizon that is longer than that of training data is considered, one-step output mapping may ignore the dependency between future predictions (e.g., between $\hat{y}_{i+1}$ and $\hat{y}_{i+2}$) and degrade the prediction accuracy. Therefore, multiple output strategy has the potential to overcome this drawback. The multiple output strategy trains one emulator to predict the forecast sequence in a single surrogate model [43]. The multiple output model is given by

$$\left[\hat{y}_{i+1}, \hat{y}_{i+2}, \cdots, \hat{y}_{i+r}\right] = f(\mathbf{u}_{i+1}, \mathbf{u}_{i+2}, \cdots, \mathbf{u}_{i+r}, y_i, y_{i-1}, \cdots, y_{i-p+1}, \boldsymbol{\theta}), \tag{7}$$

where $\hat{y}_{i+1}, \hat{y}_{i+2}, \cdots, \hat{y}_{i+r}$ are the predictions at time step $t_{i+1}, t_{i+2}, \cdots, t_{i+r}$, and r is the total number of predicted time steps. Practical applications show that multi-output models are complex and do not have enough flexibility since the stochastic dependency behavior must be learned.

3. SURROGATE MODELING FOR NONLINEAR DYNAMIC SYSTEMS

This section first discusses the generation of training data for surrogate modeling of nonlinear dynamic systems. Following that, we briefly review four surrogate modeling techniques for nonlinear dynamic systems, including GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM.

3.1 Training Data Generation for Surrogate Modeling of Nonlinear Dynamic Systems

After considering the benefits of different forecasting strategies and nonlinear dynamic structures, this paper investigates dynamic surrogate modeling for long-term prediction using recursive multi-step forecasting strategy based on short-term training data. The dynamic output is characterized by the controllable inputs and multi-step outputs of previous time steps and iterated by the single-period prediction result. All data in this paper are collected synthetically through simulations for verification and validation purpose. This paper uses the uniform time steps in the predictive model and controllable inputs. However, the time step sizes are different for different case studies.

Fig. 1 shows an overview of surrogate modeling for nonlinear dynamic systems. As shown in this figure, there are five main steps, namely: (a) generation of input training data, which generates training data for dynamic system model parameters and excitations; (b) training data collection, which obtains data of output based on simulations of dynamic systems; (c) data preprocessing, which processes the data into the format that is needed for the training of various surrogate models; (d) surrogate modeling training using the processed data; and (e) prediction and validation, which uses the trained surrogate model for prediction under new conditions and excitations. Next, we briefly explain the first three steps, and following that, we will discuss the training of various surrogate models in Secs. 3.2 through 3.5.

Place Figure 1 here

As shown in Fig. 1, training data generation is essential in training surrogate models for nonlinear dynamic systems. This paper first generates ϑ training samples for model parameters

$\boldsymbol{\theta}$ using Latin Hypercube sampling (LHS) [44]. Let the generated training samples be $\boldsymbol{\theta}_i, i = 1, 2, \cdots, \vartheta$, where $\boldsymbol{\theta}_i$ represents the *i*-th training sample. Then, for each sample $\boldsymbol{\theta}_i$, as shown in Fig. 1(b), we obtain a time-series response $y_{i,j}, j = 1, 2, \cdots, N_i$ with controllable time-series inputs $u_{i,j}, j = 1, 2, \cdots, N_i$, in which N_i is the length of the controllable input excitations of the *i*-th training sample.

After that, we process the data of inputs and outputs into training data for surrogate modeling by following the NARX scheme as follows (i.e., Fig. 1(c))

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}^{(1)} \\ \vdots \\ \mathbf{z}^{(\vartheta)} \end{bmatrix} \in \mathbb{R}^{N_T \times (p+q+s)}, \ \mathbf{Y} = \begin{bmatrix} \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(\vartheta)} \end{bmatrix} \in \mathbb{R}^{N_T \times 1},$$

where

$$\mathbf{z}^{(i)} = \begin{bmatrix} \overleftarrow{\mathbf{u}}_{i,p+1} & \overleftarrow{\mathbf{y}}_{i,p+1} & \boldsymbol{\theta}_i \\ \overleftarrow{\mathbf{u}}_{i,p+2} & \overleftarrow{\mathbf{y}}_{i,p+2} & \boldsymbol{\theta}_i \\ & \vdots & \\ \overleftarrow{\mathbf{u}}_{i,N_i} & \overleftarrow{\mathbf{y}}_{i,N_i} & \boldsymbol{\theta}_i \end{bmatrix} \in \mathbb{R}^{(N_i - p)\times(p+q+s)}, \mathbf{y}^{(i)} = \begin{bmatrix} y_{i,p+1} \\ y_{i,p+2} \\ \vdots \\ y_{i,N_i} \end{bmatrix} \in \mathbb{R}^{(N_i - p)\times 1}, \ \forall i = 1, \cdots, \vartheta$$

$$\overleftarrow{\mathbf{u}}_{i,p+m} = [u_{i,m+p}, u_{i,m+p-1}, \cdots, u_{i,m+p+1-q}] \in \mathbb{R}^{1\times q},$$

$$\overleftarrow{\mathbf{y}}_{i,p+m} = [y_{i,m+p-1}, y_{i,m+p-2}, \cdots, y_{i,m}] \in \mathbb{R}^{1\times p}, m = 1, \cdots, N_i - p \tag{8}$$

in which $N_T = \sum_{i=1}^{\vartheta} (N_i - p)$ is the total number of training points and s is the number of dimensions of $\boldsymbol{\theta}$.

The above-generated training data are then used to train various surrogate models for the nonlinear dynamic system in Secs. 3.2 through 3.5. Fig. 2 presents a single-step univariate dynamic predictive model schematic after the surrogate modeling (i.e., Fig. 1(e)). As indicated in this figure, the emulator predicts the future values; meanwhile, observed values were replaced by forecasting from the last period. Thus, for example, $\hat{y}_{i+2}$ is the predicted output in the second iteration

$(\hat{y}_{i+2} = f(u_{i+2}, u_{i+1,} \cdots, u_{i-q+3}, \hat{y}_{i+1}, y_{i,} \cdots, y_{i-p+2}) + v_{i+2})$, and the first iteration output $\hat{y}_{i+1}$ is in place of y_{i+1}. All the controllable inputs switch from the actual values to the estimated ones with the incremental recursive scheme. Meanwhile, predictions are iterated in the recursive procedure with input variables to address error accumulation to a certain extent.

Next, we will briefly review the four types of surrogate modeling techniques studied in this comparative paper, including GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM.

Place Figure 2 here

3.2 GP-NARX

3.1.1 Brief review of GP-NARX

As discussed above, a NARX model can be represented as

$$\begin{aligned} y_i &= f\left(\mathbf{x}_i, \boldsymbol{\theta}\right) + v_i, \\ \mathbf{x}_i &= \left[u_i, u_{i-1}, \cdots, u_{i-q}, y_{i-1}, y_{i-2}, \cdots, y_{i-p}\right], \end{aligned} \tag{9}$$

where $v_i \sim N(0, \sigma_v^2)$ is the white Gaussian noise following a normal distribution with zero mean and standard deviation σ_v, p and q are respectively the lags in the output and input values.

Since the nonlinear mapping function $f(\cdot)$ is usually a simulation model that requires substantial computational effort to evaluate, the direct application of the NARX model to dynamic system analysis, design, and control is computationally prohibitive. To address the computational challenge, $f(\cdot)$ can be substituted with a computationally "cheaper" surrogate model. In the GP-NARX surrogate model, the probabilistic and nonparametric Gaussian process (GP) model is employed to approximate the nonlinear mapping function $f(\cdot)$.

The response over the prediction space is assumed to be a Gaussian process in the GP model with the mean function $m(\mathbf{z})$, where $\mathbf{z} = [\mathbf{x}, \boldsymbol{\theta}] \in \mathbb{R}^{p+q+s}$ and covariance function $k(\mathbf{z}, \mathbf{z}')$ given by

$$\begin{aligned} m(\mathbf{z}) &= \mathbb{E}\left[f(\mathbf{z}) \right], \\ k(\mathbf{z}, \mathbf{z}') &= \mathbb{E}\left[\left(f(\mathbf{z}) - m(\mathbf{z}) \right) \left(f(\mathbf{z}') - m(\mathbf{z}') \right) \right]. \end{aligned} \tag{10}$$

For measured input matrix $\mathbf{Z} = \left[\mathbf{z}^{(1)}, \cdots, \mathbf{z}^{(m)} \right]^T \in \mathbb{R}^{N_T \times (p+q+s)}$, where $\mathbf{z}^{(i)}$ is the i-th input vector at the certain discrete time step of N_T discrete time steps, we can have the prior GP for function $f(\mathbf{Z})$ as

$$f(\mathbf{Z}) \sim \mathcal{GP}\left(\mathbf{m}_f, \boldsymbol{\Sigma}_f + \sigma_n^2 \mathbf{I}_{N_T} \right), \tag{11}$$

where $\mathbf{m}_f$ is the mean vector and $\boldsymbol{\Sigma}_f$ is the covariance matrix obtained by evaluating Eq. (10) for all input $\mathbf{Z}$, σ_n^2 is the variance of white noise ν_i, $\mathbf{I}_{N_T}$ is an $N_T \times N_T$ unit matrix. We can generally have $\mathbf{m}_f \equiv \mathbf{0}$, especially when we have no prior information. After observing the measured output data of $\mathbf{Z}$, i.e. $\mathcal{D} = \left\{ (\mathbf{Z}, \mathbf{y}) \right\}$, where $\mathbf{y} = \left[y_1, \cdots, y_i, \cdots, y_{N_T} \right]^T$, the posterior distribution of function $f(\mathbf{z})$ given measured data $\mathcal{D}$ and hyperparameters Θ is given by

$$p(f(\mathbf{z}) \mid \mathbf{Z}, \mathbf{y}, \Theta) = \frac{L(\mathbf{y} \mid \mathbf{Z}, f(\mathbf{z}), \Theta)\, p(f(\mathbf{z}) \mid \Theta)}{p(\mathbf{y} \mid \mathbf{Z}, \Theta)}, \tag{12}$$

where $L(\mathbf{y} \mid \mathbf{Z}, f(\mathbf{z}), \Theta)$ is the likelihood, $p(f(\mathbf{z}) \mid \Theta)$ is the prior given hyperparameters Θ, $p(\mathbf{y} \mid \mathbf{Z}, \Theta)$ is the evidence, and $p(f(\mathbf{z}) \mid \mathbf{Z}, \mathbf{y}, \Theta)$ is the posterior distribution over $f(\mathbf{z})$.

The implementation of Bayesian inference in Eq. (12) may be analytically intractable due to the evaluation of multiple integrals; one can estimate the hyperparameters Θ by maximizing the marginal likelihood [45]. After estimating the hyperparameters Θ, for any unobserved input $\mathbf{z}^*$, we have the following joint distribution.

$$\begin{pmatrix} \mathbf{y} \\ y^* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{\Sigma}_f + \sigma_n^2 \mathbf{I}_N & \mathbf{K}(\mathbf{Z}, \mathbf{z}^*) \\ \mathbf{K}(\mathbf{Z}, \mathbf{z}^*)^{\mathrm{T}} & k(\mathbf{z}^*, \mathbf{z}^*) \end{bmatrix}\right), \tag{13}$$

where $\mathbf{K}(\mathbf{Z}, \mathbf{z}^*)$ is a covariance vector between $\mathbf{z}^*$ and input vector $\mathbf{Z}$.

By transforming the joint distribution to Gaussian condition distribution $p(y^* \mid \mathcal{D}, \mathbf{z}^*, \Theta)$, we can obtain the following mean and variance prediction for $\mathbf{z}^*$

$$\begin{aligned} m(\mathbf{z}^*) &= \mathbf{K}(\mathbf{Z}, \mathbf{z}^*)^{\mathrm{T}} \left(\mathbf{\Sigma}_f + \sigma_n^2 \mathbf{I}_{N_T}\right)^{-1} \mathbf{y}, \\ \operatorname{cov}(\mathbf{z}^*, \mathbf{z}^*) &= k(\mathbf{z}^*, \mathbf{z}^*) - \mathbf{K}(\mathbf{Z}, \mathbf{z}^*)^{\mathrm{T}} \left(\mathbf{\Sigma}_f + \sigma_n^2 \mathbf{I}_{N_T}\right)^{-1} \mathbf{K}(\mathbf{Z}, \mathbf{z}^*). \end{aligned} \tag{14}$$

After the GP model is constructed, the GP-NARX model is written as

$$\begin{aligned} \hat{y}_i &= f_{GP}\left(u_i, u_{i-1}, \cdots, u_{i-q}, y_{i-1}, y_{i-2}, \cdots, y_{i-p}, \boldsymbol{\theta}\right) + \nu_i, \\ \hat{y}_{i+1} &= f_{GP}\left(u_{i+1}, u_i, \cdots, u_{i-q+1}, \hat{y}_i, y_{i-1}, \cdots, y_{i-p+1}, \boldsymbol{\theta}\right) + \nu_{i+1}. \end{aligned} \tag{15}$$

3.1.2 Sub-sampling for GP-NARX construction with a large volume of data

For high-rate time series/dynamic systems, a large volume of training data will be collected using Eq. (8). This will drastically increase the computational effort for both the training and prediction of the GP-NARX model. Aiming to address this challenge for GP-NARX surrogate modeling, this paper proposes a sub-selection method based on the max-min algorithm [46, 47]. In the max-min method, training points are selected by maximizing the minimum distance between a new point and all currently available training points as

$$\mathbf{z}_{new} = \max_{\mathbf{z} \in \mathbf{Z}} \{ \min_{\mathbf{z}^t} [\| \mathbf{z} - \mathbf{z}^t \|_2] \}, \tag{16}$$

where $\mathbf{z}^t$ denotes all currently available training points and $\|\bullet\|_2$ is the l_2-norm of a vector. Thus, this algorithm allows us to evenly fill the design domain with a small number of representative points of the original training data.

In order to sub-select N_S samples from the N_T available samples given in Eq. (8), we first randomly select N_{in} samples and denote the selected N_{in} samples as $\mathbf{z}^t = [\mathbf{z}_1^t, \mathbf{z}_2^t, \cdots, \mathbf{z}_{N_{in}}^t]$, where $\mathbf{z}_j^t$ is the *j*-th selected point. To sub-select the other $N_S - N_{in}$ samples from the $N_T - N_{in}$ available samples, we first compute the minimum distances between samples in $\mathbf{z}^t$ and samples in $\mathbf{Z}$ (i.e., Eq. (8)) as follows

$$l_k = \min_j \left\{ \left\| \mathbf{z}_j^t - \mathbf{z}(k) \right\|_2 \right\}, k = 1, 2, \cdots, N_T - N_{in}, \tag{17}$$

where $\mathbf{z}(k)$ is the *k*-th sample in $\mathbf{Z}$.

With the minimum distances obtained from Eq. (17), the index of a new sample is then identified as

$$i^* = \arg\max_k \{l_k\}. \tag{18}$$

Based on the identified index, the new sample is then added to the selected training samples $\mathbf{z}^t$. This process continues iteratively until N_S samples are sub-selected. Using the sub-selected samples, a GP-NARX surrogate model can be trained by following Eqs. (10) through (14).

3.3 Long Short-Term Memory (LSTM)

Recently, deep neural networks (DNNs), such as LSTM and CNN, have emerged as new modeling methods of temporal or sequential characteristics. It has been widely applied in various domains, such as product searching [48], advertising [49], image recognition [50], etc.

Even though LSTM, CNN-LSTM, and CNN-BLSTM belong to different types of RNNs, they resemble a mapping feedforward neural network using similar algorithms. For different databases, the models have different hyperparameters and weights in different layers. This paper employs a many-to-one RNN for prediction. Fig. 3 below shows an example of a many-to-one RNN with one hidden layer. In this figure, $a = \sigma(wx + b)$, where w and b are weights, $\sigma(\cdot)$ is the activation function (i.e., sigmoid function in this example). $[x_1, x_2, \ldots, x_{N_T}]$ are inputs to the input layer and $\hat{y}$ is the output of the output layer. As mentioned above, this paper will investigate various methods to construct accurate surrogate models for nonlinear dynamic systems. In the following, we will briefly introduce the LSTM method, which is one of the commonly used variants of RNN.

LSTM, as a variant of RNN, is widely used for solving classification and regression problems, such as PM2.5 prediction [51] and traffic flow prediction [52]. It not only performs well in capturing short-term data dependencies for prediction, but also adequately account for the explicit dependence of multiple outputs over the long-time horizon. Comparing with the traditional RNN, LSTM overcomes the shortcoming of insufficient long-term dependence in RNN due to the exploding gradient resulting from gradient propagation. Fig.4 shows the structure of an LSTM cell. As shown in this figure, in each LSTM cell, there are an input gate, a forget gate, and an output gate. The forget gate determines the information whether it should be got rid of; the input gate determines the new information, which consists of actual observed values after iterations by a dynamic model; and the output gate decides the values to the next neuron will be computed using the activation function.

Place Figures 3 and 4 here

The standard LSTM architecture comprises multiple hidden layers, which consist of input layers and output layers. For more complex LSTM models, it can add more layers such as dropout layers which can reduce the computational time. Furthermore, specific LSTM layers contain a series of LSTM cells, and each cell has its algorithm. In an LSTM network, it designs time step *t* (*t*=1, 2, …, *n*, where *n* is the number of time steps) and network layer *L* (*L*=1, 2, …, N_L, where N_L is the number of LSTM layers). Let us denote, at time step t_i, the input gate, forget gate, and output gate as i_i, f_i, and o_i, respectively, h_i as a hidden layer output and c_i as the memory cell state. The relationships among the above variables are given as follows:

$$i_i = \sigma(W_{xi} I_i + W_{hi} h_{i-1} + b_i), \tag{19}$$

$$f_i = \sigma(W_{xf} z_i + W_{hf} h_{i-1} + b_f), \tag{20}$$

$$o_i = \sigma(W_{xo} z_i + W_{ho} h_{i-1} + b_o), \tag{21}$$

$$\hat{c}_i = tanh(W_{xc} z_i + W_{hc} h_{i-1} + b_c), \tag{22}$$

$$c_i = f_i \Lambda c_{i-1} + i_i \Lambda \hat{c}_i, \tag{23}$$

$$h_i = o_i \Lambda \tanh(c_i), \tag{24}$$

where Λ represents the element-wise process operation (Hadamard product), tanh represents the hyperbolic tangent function, *W* with two subtitles matrixes are weights between two gates, $W_{\alpha\beta}$ is the weight with different gates, b_β is the bias vector, where $\alpha = \{X, h\}$ and $\beta = \{f, i, c, o\}$. For example, $W_{hf}^{(l)}$ represents the *l*-th layer of LSTM's weight matrices equaling to the input vector h_i within the forget gate f_i; $b_f^{(l)}$ denotes the *l*-th layer of LSTM's bias vector corresponding to the forget gate.

For the first step, in Eq. (20), the input data $\{z_i^l, h_{i-1}^l\}$ goes through the forget gate (σ sigmoid function), and it discharges a vector with the value of 0 and 1 to determine the latter information. It forgets the value c_{i-1}^l when the vector value is 0, while passing c_{i-1}^l value when the vector value is 1. Then the next step determines what information can be added to $\hat{c}_i$ by using the hyperbolic tangent function in Eq. (22). Afterward, in Eq. (23) the elementwise Hadamard product function combines $\hat{c}_i$ and i_i to update the new memory state outputs c_i^l. Lastly, this single LSTM cell's outputs h_i^l can be obtained through Eq. (24) using the Hadamard product function with updated memory outputs c_i^l (scaled within value between -1 and 1) and output gate results o_i^l.

Fig. 5 depicts a three-layer LSTM architecture. In the three layers of Fig. 5, the input states are denoted as $z = [x_1, x_2, ..., x_j, ..., x_{N_T}]$, and the output is $\hat{y}$, where *m* is the total number of input or output datasets. *p*₁, *p*₂, *p*₃ are respectively the neural number of layers one, two, and three. In each LSTM layer, LSTM cell output $\left\{c_i^l, h_i^l\right\}$ passes through p_l nodes according to the LSTM cell algorithms, and output y_i^l equals the *l*-th hidden state response h_i^l, which is transferred to the next LSTM layer for the input state. Finally, the fully connected (FC) layer connects the LSTM and output layers to obtain output features.

Place Figure 5 here

Besides, the dropout layers can be added after each RNN layer to reduce the training time and avoid overfitting. The main idea of the dropout is to exclude input and nodes with a certain probability randomly. Fig. 6 (a) and (b) illustrate a standard neural network and the network after dropout.

Place Figure 6 here

3.4 Convolutional Neural Network Combined with Long Short-Term Memory (CNN-LSTM)

CNN-LSTM is a hybrid model combines two different deep learning models, which integrates large data features and even spatial data structures with more than two dimensions. CNN is designed to perform sequential predictions using inputs like pictures and videos which cannot directly be modeled by the standard LSTM [53]. In CNN-LSTM, the jointly constructed model by CNN and LSTM works in three steps to perform dynamic surrogate modeling. For CNN, it can extract features of dynamic models. Then features collected by CNN are transferred to the following layers, such as LSTM layers or deeper LSTM network for bidirectional feature learning (BLSTM) in the forward and backward directions. Then the dense layer is used to represent the features and transfer them to the prediction layer.

Unlike the standard RNN, CNNs are constructed with neurons like LSTM neurons that can capture the features through learnable biases and weights. As shown in Fig. 7, the difference between CNN with traditional ANN is that the input of CNN can be three dimensions: width, height, and depth, which are widely used when images are inputs. In this paper, CNN transfers the inputs from one dimension into two or three dimensions to get a higher prediction accuracy. Moreover, the final output layer of CNN will reduce the multiple vectors into a single vector, all the temporal features of the dynamic models can be preserved for the following neural networks (i.e., LSTM).

Place Figure 7 here

Fig. 8 shows an example of a CNN-LSTM architecture. The input state for the model is $\mathbf{Z} \in \mathbb{R}^{1\times(p+q+s)}$. O filters $\{W_1, W_2, ..., W_O\}$ are implemented in convolution operations. The process of convolution learning is written as

$$WL = \varpi(\mathbf{Z} * \{W_1, W_2, ..., W_O\} + b_c), \tag{25}$$

where *WL* is the feature map setting in the convolutional layer, ϖ is the activation function (ReLU function with max (0, *x*)), b_c is the bias of the convolutional layer, * represents the convolutional operation.

Then all the collected convolutional features maps $WL \in \mathbb{R}^{1\times\kappa}$ (through κ time steps) are transferred into LSTM layers, which has been discussed in Sec. 3.3. As shown in Fig. 8, a CNN model is adopted for feature extraction in the first half part, while an LSTM model is adopted for prediction in the other half part with the extracted dataset from the CNN layers as inputs. Through this hybrid structure, the prediction for the current step will be added into the observed values as part of the following control input according to the recursive prediction scheme discussed in Sec. 2 to perform long-term prediction for the future time steps.

Place Figure 8 here

3.5 CNN-Bidirectional Long Short-Term Memory (CNN-BLSTM)

Bidirectional long short-term memory (BLSTM) is an extension of traditional LSTM. Instead of having only one forward direction like LSTM, it establishes two training directions, as shown in Fig. 9, where one is in the input order, and the other one is in a reversed order of the initial one.

Fig. 10 illustrates the structure of CNN-BLSTM. The only difference between CNN-LSTM and CNN-BLSTM is to replace LSTM with bidirectional LSTM. In this paper, two bidirectional LSTM layers are stacked together to get the hidden gate output as below:

$$h_t^l = \begin{cases} \rho(\iota(WL, \vec{h}_{t-1}^l), \iota(WL, \overleftarrow{h}_{t-1}^l)), & l = 1 \\ \rho(\iota(h_{t-1}^{l-1}, \vec{h}_{t-1}^l), \iota(h_{t-1}^{l-1}, \overleftarrow{h}_{t+1}^l)), & l = 2 \end{cases}, \tag{26}$$

where h_t^l is the hidden gate output of the *l*-th bidirectional response at the time step *t*, $\vec{h}$ is forward direction of bidirectional layer, $\overleftarrow{h}$ is backward direction of bidirectional layer, ι is the LSTM cell (from Eq.(16) to (21)), and ρ is the function of combined forward ($\vec{h}$) and backward ($\overleftarrow{h}$) direction calculation sequences.

Place Figures 9 and 10 here

The surrogate model constructed using CNN-BLSTM is similar to that from CNN-LSTM.

3.6 Summary of Models and Implementation Procedure

In this paper, all the above four models are implemented in Python environment. For GP-NARX, the model is developed based on *Scikit-learn* package. For LSTM, CNN-LSTM, and CNN-BLSTM, the models are implemented using *Tensorflow* and *Keras* packages. In the spirit of the "Occam's Razor" principle, when presented with competing models that perform with similar predictive power, the 'optimal' model should be the one with the least complexity, by some measure, which we interpret to mean tunable parameters for the purposes of this study. To this end, we summarize the tunable parameters, advantages, and disadvantages of GP-NARX and deep learning-based models in Table 1 for the selection of models.

Place Table 1 here

Next, we will use three numerical examples to perform a comprehensive comparative study of the aforementioned surrogate modeling techniques.

4. Comparative Studies

Three examples are used in this section to compare the performance of different surrogate modeling methods. The first one is a mathematical example with a two-step lag input and a one-step-ahead prediction. The second one is a duffing oscillator model. The last one is the Bouc-Wen nonlinear dynamic model. The exemplar complexity increases from Example 1 to Example 3.

In the past decades, various approaches have been developed to check the accuracy of dynamic model prediction, such as mean square error (MSE), correlation and median [54], probabilistic error measure using reliability theory [55], and dynamic time warping [56]. Since MSE is the most widely used in surrogate modeling, MSE as below is adopted in this paper to measure the prediction accuracy of different surrogate modeling methods

$$MSE = \frac{1}{N_d} \sum_{i=1}^{N_d} (y_i - \hat{y}_i)^2, \tag{27}$$

where N_d is the total number of test data, y_i and $\hat{y}_i, \forall i = 1, \cdots, N_d$, are respectively the observed and predicted values.

In addition, the deep learning models (i.e. LSTM, CNN-LSTM, and CNN-BLSTM) in all three examples are tuned to achieve the best performance. The dropout rate is tuned in the range of [0.001, 0.1] by comparing the performance of four different dropout rates, 0.001, 0.005, 0.01, and

0.1. Four different batch sizes (16, 32, 64, and 128) are also compared to determine the best batch size for different examples. The performances of LSTM models with different number of layers (2, 3, 4, and 5) are also compared in each example to select the best number of layers. The learning rate is tuned using 0.0001, 0.001, 0.01, and 0.1. The number of neurons is tuned with 30, 40, 60, 80, and 100 to select the optimal number of neurons. For the CNN model, two layers are used for convolution and pooling. The pooling size for the CNN layer is selected as 2 after comparing the performance of three different sizes, namely 2, 3, and 6. Even though the hyperparameters of the deep learning models might be different for different problems, the optimal hyperparameters for the three studied problems turn out to be the same, which are: dropout rate-0.01, batch size-32, number of hidden layers-3, learning rate-0.001, and number of neurons-80. The number of epochs is also tuned for each example to ensure the convergence of the loss functions. The required tunning effort for the deep learning models in general is much higher than that of GP-NARX.

4.1 Example 1: A Mathematical Example

A mathematical nonlinear dynamic predictive model is given by

$$y_i = \cos\left(\frac{y_{i-1}}{y_{i-2}+2}\right) + 0.8\sin(y_{i-2}^2 + y_{i-1}) + v_i, \tag{28}$$

where v_i is noise and follows the normal distribution $v_i \sim N(0, \sigma_v^2)$, $\sigma_v = 0.1$, and y_i is the response at the *i*-th time step.

As shown in Eq. (28), it is a two-step lag nonlinear dynamic model. Since the input only includes outputs from previous time steps, it can be considered as a one-dimensional NARX model. In order to compare the performance of the four different surrogate modeling methods, the predictive model given in Eq. (28) is assumed to be unknown. Therefore, surrogate models are constructed using training data generated using the predictive model. In this case study, five trajectories of time series data are generated using Eq. (28) as the training data. The first two-time step responses of the five-time series are respectively $[0.6, 1.2]$, $[1.2, -0.6]$, $[0, 0]$, $[-1.2, -1.2]$, and $[-0.6, 0.6]$. The time series lengths are 20, 50, 50, 30, and 50, respectively. Through the five training time series, 195 training samples are obtained. Fig. 11 presents the five-time series training data.

Based on the training data given in Fig. 11, four surrogate models are constructed using GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM, respectively, following the methods discussed in Sec. 3. Two scenarios, namely *with subsampling* and *without subsampling*, are considered to compare the performance of different surrogate models. For surrogate modeling with subsampling, 150 training points are sub-selected from the available data. Since the total number of training data is small, it allows us to train the GP-NARX model using all the training data for the scenario without subsampling. In order to quantitatively quantify the accuracy of different surrogate models, Table 2 gives the MSE comparison of the four surrogate models with subsampling for 14 testing cases. Following that, Table 3 lists the MSE comparison for the scenario of surrogate modeling without subsampling.

Place Figure 11 here

Place Table 2 here

Comparing the results in Tables 2 and 3, it shows that increasing the number of training data in general can increase the accuracy of all four types of surrogate models. Since GP-NARX model can be trained using all the training data, we focus on the results in Table 3 for the comparative study. The results in Table 3 indicates that GP-NARX has a higher prediction accuracy than the RNN models (i.e., LSTM, CNN-LSTM, CNN-BLSTM) in general, when the period of dynamic prediction is long. For example, the MSE of GP-NARX prediction is 0.0038 while its counterpart of the RNN models is over 0.01 when the prediction period is 200-time steps (i.e., Case 8). When the period for prediction is 40-time steps (i.e., Case 1), the MSE of GP-NARX is over 0.03 while that of RNN models is less than 0.015, as shown in Table 3.

Fig. 12 presents the comparison of surrogate model prediction and true response for one testing case for the scenario of surrogate modeling without subsampling. Following that, Fig. 13 gives the corresponding prediction errors of the four surrogate models for the testing case. The results plotted in Figs. 12 and 13 show that the predictive model constructed using LSTM has the highest prediction accuracy for this particular testing case.

Place Table 3 here

It can be concluded from this particular mathematical example that the GP-NARX constructed based on the time series given in Fig. 11 is more suitable than RNN models for the prediction of long periods. On the other hand, RNN models, including LSTM, CNN-LSTM, and CNN-BLSTM models, are more suitable for predictions that have a similar data shape or distribution as the training data. For instance, the lengths of the training time series are between 20 to 50, and the RNN models have a higher prediction accuracy than GP-NARX when the number of prediction time steps is close to that range. In addition, the results in Table 3 show that the prediction capability of RNN models is more robust than that of GP-NARX for this particular example, which is manifested as a lower average MSE value and a more stable performance.

Place Figure 12 here

Place Figure 13 here

4.2 Example 2: An Asymmetric Duffing Oscillator

An asymmetric duffing oscillator is adopted from Ref. [24] as our second example. It is a nonlinear second-order differential equation used to model certain damped and driven oscillator, which exhibits chaotic behavior and describes the dynamics of a point mass in a double well potential [57]. The equations of motion are given by

$$\begin{aligned} &\upsilon\ddot{y} + c\dot{y} + ky + k_2 y^2 + k_3 y^3 = u(t), \\ &y_{i(obs)} = y_i + v_i, v_i \sim N(0, \sigma_v^2), \end{aligned} \tag{29}$$

in which $u(t)$ is the input excitation given by

$$u(t) = a \times cos(bt) + \left(sin\left((b+3)t\right) + 2\right) + 2a \times cos(bt+1) + sin(2bt), \tag{30}$$

where $\sigma_v = 2\times10^{-6}$, υ =1, c=10, k=2×10^4, k_2=10^7, and k_3=5×10^9. For demonstration purpose, *a, b*, and *c* are treated as controllable model parameters that can be changed for different experiments, and thus we have $\boldsymbol{\theta} = [a, b, c]$.

In order to generate training data for the surrogate models, we first generate 60 training data for $\boldsymbol{\theta}$ with a lower bound $\boldsymbol{\theta}_L \in \{2, 0.1, 0.1\}$ and an upper bound $\boldsymbol{\theta}_U \in \{25, 10, 15\}$ using the Latin Hypercube sampling method, and we have ϑ = 60 in Eq. (8). The range of $\boldsymbol{\theta}$ is only used for demonstration purpose in this paper. In probabilistic analysis or design of nonlinear dynamic systems, the range needs to be determined according to distributions of $\boldsymbol{\theta}$. For each training dataset of $\boldsymbol{\theta}$, excitations of $u(t)$ are generated after adding noise to Eq. (30) based on the values of a and b. The length of each u(t) is around 2000, and therefore we have $N_i = \{n_i \mid 1200 \leq n_i \leq 2500\}$, $\forall i = 1, 2, \cdots, 60$ in Eq. (29). For each excitation and training data of c, we solve Eq. (29) using the fourth-order fixed-step Runge-Kutta algorithm and obtain the responses of y. Fig. 14 shows one example of the 60 training excitations corresponding responses of y.

Place Figure 14 here

Through cross-validation and after comparing the performance of surrogate models with different numbers of lag ranging from 5 to 15, it is found that a time lag of nine for both input excitation $u(t)$ and output $y(t)$ gives the best prediction accuracy in surrogate modeling for all four types of surrogate models. We, therefore, have p=9 and q=9. Following the procedure discussed in Sec. 3.1, we obtain training data for surrogate modeling. Through the 60 training time series, 100211 training samples are obtained. The large volume of training data makes the training of GP-NARX computationally very challenging. We, therefore, perform subsampling of the training data using the approach discussed in Sec. 3.1.2. From the subsampling, 900 training data are sub-selected for the training of GP-NARX while maintaining the space-filling property. Fig. 16 shows a comparison between the sub-selected samples and the original samples for two dimensions of the training data (i.e., u_1 *v.s.* y_1 and u_3 *v.s.* y_3). As shown in this figure, the subsampling method can well-maintain the space-filling property while reducing the number of training data.

Place Figure 15 here

The sub-selected training data are used to train GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM models. Additionally, another set of LSTM, CNN-LSTM, and CNN-BLSTM models are trained using all the training data. We therefore have one GP-NARX model trained based on subsampling, two sets of LSTM, CNN-LSTM, and CNN-BLSTM models respectively trained with and without subsampling. After training the surrogate models, we randomly generate eight samples for $\boldsymbol{\theta} = [a, b, c]$ according to the lower and upper bounds. Based on that, eight random input excitations are obtained using Eq. (30). We then assume that the original dynamic model is unknown and compare predictions of surrogate models with responses of the original dynamic model for the eight testing datasets. Since the results of LSTM, CNN-LSTM, and CNN-BLSTM models with subsampling are much worse than that of deep learning models without subsampling, GP-NARX with subsampling is directly compared with LSTM, CNN-LSTM, and CNN-BLSTM models trained using all available training data. Table 4 gives the MSE comparison of the four surrogate models for all eight testing cases. The average MSE for these eight-test datasets is 2.29×10^{-08} for GP-NARX, the best among the four studied surrogate models. In addition, the deep learning models (LSTM, CNN-LSTM, and CNN-BLSTM) have similar performances. Thus, the results in Table 4 show that GP-NARX has a higher prediction accuracy than the other methods for this particular example. In addition, GP-NARX has a stable performance when the data changes drastically over long periods.

Place Table 4 here

Figs.16 presents the comparison of predictions and the true responses for one of the eight testing cases. Following that, Fig.17 shows the corresponding kernel density estimates of error distribution for the four surrogate models. The results in Figs. 16 and 17 show that the predictive model constructed using GP-NARX has the highest prediction accuracy for this testing case (i.e. Case 1 in Table 4).

Place Figures 16 and 17 here

4.3 Example 3: The Bouc-Wen Model

In structural engineering, many nonlinear inelastic material behaviors are of interest in structures such as reinforced concrete, steel, base isolation systems, damping devices, etc [58]The Bouc-Wen model, proposed by [59, 60], and extended by [61], is used in this work as the third example due to its flexibility to capture the behavior of many inelastic material models by just changing its tunable parameters.

There are different variants of the Bouc-Wen model. However, the model and notation shown in [62] have been adopted in this work. To describe this model, consider the single degree of freedom (SDOF) system:

$$\upsilon\ddot{x}(t)+\psi\dot{x}(t)+\underbrace{F(\dot{x}(t),x(t))}_{F(t)}=\alpha(t), \tag{31}$$

where $x(t)$, $\dot{x}(t)$, $\ddot{x}(t)$ are the displacement, velocity, and acceleration response of the SDOF system, respectively. Also, υ, ψ, and $\alpha(t)$ represent the mass, damping coefficient, and excitation force, respectively. The term $F(t)$ is the restoring force, which according to the Bouc-Wen Model, can be expressed as follows:

$$F(t) = \eta k_i u(t) + (1-\eta) k_i z(t), \tag{32}$$

where η denotes the ratio between the post-yield stiffness and the pre-yield stiffness (i.e. k_i) and $z(t)$ denotes a non-observable hysteretic displacement. As noted, Eq (32) is modeled as the sum of a linear and a nonlinear function.

The nonlinear function is obtained by solving the following nonlinear differential equation

$$\dot{z}(t) = \dot{x}(t) - \beta \left|\dot{x}(t)\right| \left|z(t)\right|^{\varsigma-1} z(t) - \gamma \dot{x}(t) \left|z(t)\right|^{\varsigma}, \tag{33}$$

which may be solved by using iterative methods to approximate its solution. The variables β, γ and ς are the Bouc-Wen tunable parameters that are used to model several different materials. For more details on applying the Bouc-Wen Model to an MDOF, the reader can refer to [4]. For this work, these values are set as constants. Fig. 18 shows an example of input excitation and corresponding dynamic responses for a two-story building nonlinear dynamic system.

In this example, we first generate 100 random excitations as the training time series. Based on the training excitations, 29011 training samples are obtained. Using the subsampling approach discussed in Sec. 3.1.2, 2000 training data are sub-selected. Similar as that in Example 2, a GP-NARX model is trained using the sub-selected training data and two sets of LSTM, CNN-LSTM, and CNN-BLSTM models are respectively trained with the sub-selected data and with all available training data. The performance of LSTM, CNN-LSTM, and CNN-BLSTM models trained with subsampling is also much worse than that trained with all available data. We therefore also only compare GP-NARX trained using sub-selected data with deep learning models trained using all available data.

Place Figure 18 here

Tables 5 and 6 give the MSE comparison of the four surrogate models for all eight testing cases for the drifts of degree of freedom (DOF) 1 and 2. The average MSE of GP-NARX for the eight test datasets is 0.011 and 0.017 respectively for the drifts of the two DOFs. It implies that GP-NARX is the best among the four studied surrogate models for this particular example based on this group of training data. In addition, LSTM has the best performance for test case 1 while GP-NARX has a better overall performance than the other methods.

Place Tables 5 and 6 here

Fig. 19 gives the random input excitation of a case study that is used to verify the accuracy of various surrogate models. Following that, Figs. 20 and 21 present the comparison of surrogate model prediction and the true response (i.e., force-displacement hysteresis) for the testing case (i.e., Case 4 in Tables 3 and 4). Figs. 22-25 show the comparison of drifts from surrogate model prediction and the true response. To more intuitive compare the accuracy of different methods, Fig. 26 depicts the prediction error distributions of drifts for the four surrogate models.

Place Figures 19-26 here

The results in Figs. 20 through 26 show that the predictive model constructed using GP-NARX has the highest prediction accuracy for this particular testing case. Amongst the three deep learning methods, LSTM has the best performance while CNN-LSTM and CNN-BLSTM show similar performances.

5. Discussion and conclusion

This paper performs a comparative study for surrogate modeling of nonlinear dynamic systems using four types of surrogate models, namely GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM. The performances of different surrogate models for multistep-ahead prediction are compared using three examples. The results show that (1) GP-NARX in general performs better than the other three types of surrogate modeling methods for the three particular examples; (2) the performance of GP-NARX is also relatively more robust than the other methods; (3) for some cases, deep learning-based surrogate modeling methods (i.e., LSTM, CNN-LSTM, and CNN-BLSTM) perform better than GP-NARX. But the required tuning effort of deep learning-based methods is higher than that of GP-NARX; (4) deep learning-based surrogate modeling methods are more suitable for prediction that has a data shape which is similar as the training datasets; and (5) all the studied surrogate modeling methods can perform long-term multi-step prediction based on training data of short periods. It also demonstrates the promising potential of various machine learning methods for surrogate modeling of nonlinear dynamic systems.

Even though deep learning methods are becoming more and more popular in many fields, they do not provide a universal solution to all problems. For instance, deep learning-based methods are better than the conventional GP-NARX for several testing cases in the first example. But GP-NARX in general performs better than deep learning-based methods in the second and third examples. It is not suggested to substitute conventional machine learning methods with deep learning methods for surrogate modeling in various engineering applications without detailed investigations. It is therefore suggested that for problems that can be solved using conventional methods such as GP-NARX, following the “Occam’s Razor” principle, we should employ such a method, since the required tuning effort of conventional methods is much lower. Deep learning methods (i.e., LSTM, CNN-LSTM, and CNN-BLSTM) have a significant advantage over GP-NARX in dealing with big data, such as videos, images, and high-dimensional data. Even for big data problems, GP-NARX method is still worth investigating after using dimension reduction techniques, such as autoencoder, or using subsampling method as what been discussed in this paper. The guidance of choosing surrogate models is to always start from the classical GP-NARX models whenever it is applicable, since it tends to be more robust and easier to tune than deep learning methods. In some situations, deep learning models can be used in conjunction with GP-NARX to improve the predictive capability. For instance, a CNN model can be used to reduce the high-dimensional problems into low-dimensional ones in the latent space, within which GP-NARX models can be employed to capture of temporal variability over time.

This paper only compared the performance of various surrogate modeling methods for deterministic predictions. Since the confidence of dynamic prediction is usually of interest to decision makers, the capability of various methods in providing probabilistic prediction is worth investigating in future. In addition, the quality of surrogate models is affected by the data used for training, how to collect the most effective data for training surrogate models of nonlinear dynamic systems is also a very interesting research topic to be further studied.

Acknowledgment

The US Army Corps of Engineers provided funding for this work through the U.S. Army Engineer Research and Development Center Research Cooperative Agreement W912HZ-17-2-0024. The support is gratefully acknowledged. LA-UR-21-31231 Approved for public release; distribution is unlimited.

References

[1] Irizarry, R., 2005, "A generalized framework for solving dynamic optimization problems using the artificial chemical process paradigm: Applications to particulate processes and discrete dynamic systems," Chemical Engineering Science, 60(21), pp. 5663-5681.

[2] Xu, B., He, J., and Masri, S. F., 2015, "Data-Based Model-Free Hysteretic Restoring Force and Mass Identification for Dynamic Systems," Computer-Aided Civil and Infrastructure Engineering, 30(1), pp. 2-18.

[3] Pravin, P., Bhartiya, S., and Gudi, R. D., 2019, "Modeling and Predictive Control of an Integrated Reformer–Membrane–Fuel Cell–Battery Hybrid Dynamic System," Industrial & Engineering Chemistry Research, 58(26), pp. 11392-11406.

[4] Tahmasian, S., 2021, "Dynamic analysis and optimal control of drag-based vibratory systems using averaging," Nonlinear Dynamics , 104(3), 2201-2217.

[5] Li, M., and Lai, A. C. K., 2015, "Review of analytical models for heat transfer by vertical ground heat exchangers (GHEs): A perspective of time and space scales," Applied Energy, 151, pp. 178-191.

[6] Butt, W. A., Yan, L., and Amezquita S, K., 2013, "Adaptive integral dynamic surface control of a hypersonic flight vehicle," International Journal of Systems Science, 46(10), pp. 1717-1728.

[7] Yoon, S., 2003, "A Study on Terrain-Surface Modeling and Searching Algorithms for Real-time Simulation of Off-Road Vehicles," Vehicle System Dynamics, 39(5), pp. 353-363.

[8] Ouyang, M., 2014, "Review on modeling and simulation of interdependent critical infrastructure systems," Reliability Engineering & System Safety, 121, pp. 43-60.

[9] Gerdes, J., Bruck, H. A., and Gupta, S. K., 2019, "A simulation-based approach to modeling component interactions during design of flapping wing aerial vehicles," International Journal of Micro Air Vehicles, 11 , 1756829318822325.
[10] Tavecchia, G., Miranda, M. A., Borras, D., Bengoa, M., Barcelo, C., Paredes-Esquivel, C., and Schwarz, C., 2017, "Modelling the range expansion of the Tiger mosquito in a Mediterranean Island accounting for imperfect detection," Front Zool, 14, p. 39.
[11] Thomas, N., Portyankina, G., Hansen, C. J., and Pommerol, A., 2011, "HiRISE observations of gas sublimation-driven activity in Mars' southern polar regions: IV. Fluid dynamics models of CO2 jets," Icarus, 212(1), pp. 66-85.
[12] Rivals, I., & Personnaz, L. , 1996, "Black-box modeling with state-space neural networks.," Neural Adaptive Control Technology pp. 237-264.
[13] Pillonetto, G., and De Nicolao, G., 2010, "A new kernel-based approach for linear system identification," Automatica, 46(1), pp. 81-93.
[14] Hong, X., Mitchell, R. J., Chen, S., Harris, C. J., Li, K., and Irwin, G. W., 2008, "Model selection approaches for non-linear system identification: a review," International Journal of Systems Science, 39(10), pp. 925-946.
[15] Masti, D., and Bemporad, A., 2021, "Learning nonlinear state–space models using autoencoders," Automatica, 129, p. 109666.
[16] Deshmukh, A. P., and Allison, J. T., 2017, "Design of Dynamic Systems Using Surrogate Models of Derivative Functions," Journal of Mechanical Design, 139(10) , 101402.
[17] Gedon, D., Wahlström, N., Schön, T. B., and Ljung, L., 2021, "Deep State Space Models for Nonlinear System Identification," IFAC-PapersOnLine, 54(7), pp. 481-486.
[18] Schön, T. B., Wills, A., and Ninness, B., 2011, "System identification of nonlinear state-space models," Automatica, 47(1), pp. 39-49.
[19] Van Der Voort, M., Dougherty, M., and Watson, S., 1996, "Combining kohonen maps with arima time series models to forecast traffic flow," Transportation research. Part C, Emerging technologies, 4(5), pp. 307-318.
[20] Ing, C.-K., and Wei, C.-Z., 2005, "Order Selection for Same-Realization Predictions in Autoregressive Processes," The Annals of Statistics, 33(5), pp. 2423-2474.
[21] Li, D., 2012, "A note on moving-average models with feedback," Journal of Time Series Analysis, 33(6), pp. 873-879.

[22] Chang, S., Wei, X., Su, F., Liu, C., Yi, G., Wang, J., Han, C., and Che, Y., 2020, "Model Predictive Control for Seizure Suppression Based on Nonlinear Auto-Regressive Moving-Average Volterra Model," IEEE Transactions on Neural Systems and Rehabilitation Engineering, 28(10), pp. 2173-2183.

[23] Mustapa, R. F., Dahlan, N. Y., Yassin, A. I., and Nordin, A. H., 2020, "Quantification of energy savings from an awareness program using NARX-ANN in an educational building," Energy and Buildings, 215, p. 109899.

[24] Worden, K., Becker, W. E., Rogers, T. J., and Cross, E. J., 2018, "On the confidence bounds of Gaussian process NARX models and their higher-order frequency response functions," Mechanical Systems and Signal Processing, 104, pp. 188-223.

[25] Zhang, Y., Kimberg, D. Y., Coslett, H. B., Schwartz, M. F., and Wang, Z., 2014, "Multivariate lesion-symptom mapping using support vector regression," Human brain mapping, 35(12), pp. 5861-5876.

[26] Tao, J.-W., and Wang, S.-T., 2012, "Kernel support vector machine for domain adaptation," Zi dong hua xue bao, 38(5), pp. 797-811.

[27] Papadopoulos, S., Azar, E., Woon, W.-L., and Kontokosta, C. E., 2017, "Evaluation of tree-based ensemble learning algorithms for building energy performance estimation," Journal of Building Performance Simulation, 11(3), pp. 322-332.

[28] Chen, W., Zhang, S., Li, R., and Shahabi, H., 2018, "Performance evaluation of the GIS-based data mining techniques of best-first decision tree, random forest, and naïve Bayes tree for landslide susceptibility modeling," Science of The Total Environment, 644, pp. 1006-1018.

[29] Mantas, C. J., Castellano, J. G., Moral-García, S., and Abellán, J., 2019, "A comparison of random forest based algorithms: random credal random forest versus oblique random forest," Soft Computing, 23(21), pp. 10739-10754.

[30] Loukeris, N., and Eleftheriadis, I., 2015, "Further Higher Moments in Portfolio Selection and A Priori Detection of Bankruptcy, Under Multi-layer Perceptron Neural Networks, Hybrid Neuro-genetic MLPs, and the Voted Perceptron," International journal of finance and economics, 20(4), pp. 341-361.

[31] Botvinick, M. M., and Plaut, D. C., 2006, "Short-term memory for serial order: A recurrent neural network model," Psychological Review, 113(2), pp. 201-233.

[32] Song, X., Liu, Y., Xue, L., Wang, J., Zhang, J., Wang, J., Jiang, L., and Cheng, Z., 2020, "Time-series well performance prediction based on Long Short-Term Memory (LSTM) neural network model," Journal of Petroleum Science and Engineering, 186, p. 106682.

[33] Chen, H., Zhang, Y., Kalra, M. K., Lin, F., Chen, Y., Liao, P., Zhou, J., and Wang, G., 2017, "Low-Dose CT With a Residual Encoder-Decoder Convolutional Neural Network," IEEE Transactions on Medical Imaging, 36(12), pp. 2524-2535.

[34] Pak, U., Kim, C., Ryu, U., Sok, K., and Pak, S., 2018, "A hybrid model based on convolutional neural networks and long short-term memory for ozone concentration prediction," Air Quality, Atmosphere, & Health, 11(8), pp. 883-895.

[35] Filiński, M., Wachel, P., and Tiels, K., "Low-Dimensional Decompositions for Nonlinear Finite Impulse Response Modeling," Springer International Publishing, pp. 352-359.

[36] Shokry, A., Baraldi, P., Zio, E., and Espuña, A., 2020, "Dynamic Surrogate Modeling for Multistep-ahead Prediction of Multivariate Nonlinear Chemical Processes," Industrial & Engineering Chemistry Research, 59(35), pp. 15634-15655.

[37] Piga, D., and Tóth, R., 2014, "A bias-corrected estimator for nonlinear systems with output-error type model structures," Automatica, 50(9), pp. 2373-2380.

[38] Rahrooh, A., and Shepard, S., 2009, "Identification of nonlinear systems using NARMAX model," Nonlinear Analysis: Theory, Methods & Applications, 71(12), pp. e1198-e1202.

[39] ElSaid, A., El Jamiy, F., Higgins, J., Wild, B., and Desell, T., 2018, "Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration," Applied Soft Computing, 73, pp. 969-991.

[40] Xue, P., Jiang, Y., Zhou, Z., Chen, X., Fang, X., and Liu, J., 2019, "Multi-step ahead forecasting of heat load in district heating systems using machine learning algorithms," Energy, 188.

[41] Zhan, X., Zhang, S., Szeto, W. Y., and Chen, X., 2019, "Multi-step-ahead traffic speed forecasting using multi-output gradient boosting regression tree," Journal of Intelligent Transportation Systems, 24(2), pp. 125-141.

[42] Wang, J., Jiang, H., Wu, Y., and Dong, Y., 2015, "Forecasting solar radiation using an optimized hybrid model by Cuckoo Search algorithm," Energy, 81, pp. 627-644.

[43] Brownlee, J., March 3, 2017, "4 Strategies for Multi-Step Time Series Forecasting," https://machinelearningmastery.com/multi-step-time-series-forecasting/.

[44] Yu, H., Chung, C. Y., Wong, K. P., Lee, H. W., and Zhang, J. H., 2009, "Probabilistic Load Flow Evaluation With Hybrid Latin Hypercube Sampling and Cholesky Decomposition," IEEE Transactions on Power Systems, 24(2), pp. 661-667.
[45] Rasmussen, C. E., and Williams, C. K., 2006, Gaussian processes for machine learning, MIT press Cambridge, MA.
[46] Johnson, M. E., Moore, L. M., and Ylvisaker, D., 1990, "Minimax and maximin distance designs," Journal of Statistical Planning and Inference, 26(2), pp. 131-148.
[47] Hu, Z., Hu, C., Mourelatos, Z. P., and Mahadevan, S., 2019, "Model Discrepancy Quantification in Simulation-Based Design of Dynamical Systems," Journal of Mechanical Design, 141(1), 011401.
[48] Lee, H. I., Choi, I. Y., Moon, H. S., and Kim, J. K., 2020, "A Multi-Period Product Recommender System in Online Food Market based on Recurrent Neural Networks," Sustainability (Basel, Switzerland), 12(3), p. 969.
[49] Zhou, L., 2020, "Product advertising recommendation in e-commerce based on deep learning and distributed expression," Electronic commerce research, 20(2), pp. 321-342.
[50] Jiang, X., Sun, J., Li, C., and Ding, H., 2018, "Video Image Defogging Recognition Based on Recurrent Neural Network," IEEE transactions on industrial informatics, 14(7), pp. 3281-3288.
[51] Li, S., Xie, G., Ren, J., Guo, L., Yang, Y., and Xu, X., 2020, "Urban PM 2.5 Concentration Prediction via Attention-Based CNN–LSTM," Applied Sciences, 10(6), p. 1953.
[52] Tian, Y., Zhang, K., Li, J., Lin, X., and Yang, B., 2018, "LSTM-based traffic flow prediction with missing data," Neurocomputing, 318, pp. 297-305.
[53] Li, Y., Su, H., Qi, C. R., Fish, N., Cohen-Or, D., and Guibas, L. J., 2015, "Joint embeddings of shapes and images via CNN image purification," ACM transactions on graphics (TOG), 34(6), 1-12.
[54] Sarin, H., Kokkolaras, M., Hulbert, G., Papalambros, P., Barbat, S., and Yang, R.-J., 2008, "A comprehensive metric for comparing time histories in validation of simulation models with emphasis on vehicle safety applications," Proc. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, August 3–6, 2008, Brooklyn, New York, USA, pp. 1275-1286.
[55] Ao, D., Hu, Z., Mahadevan, S., 2017, "Dynamics model validation using time-domain metrics," Journal of Verification, Validation and Uncertainty Quantification, 2(1), 011004.

[56] Sarin, H., Kokkolaras, M., Hulbert, G., Papalambros, P., Barbat, S., and Yang, R.-J., 2010, "Comparing time histories for validation of simulation models: error measures and metrics," Journal of Dynamic Systems, Measurement, and Control 132(6), 061401.
[57] Wiggins, S., 1987, "Chaos in the quasiperiodically forced duffing oscillator," Physics Letters A, 124(3), pp. 138-142.
[58] Pei, J.-S., Gay-Balmaz, F., Luscher, D. J., Beck, J. L., Todd, M. D., Wright, J. P., Qiao, Y., Quadrelli, M. B., Farrar, C. R., and Lieven, N. A., 2021, "Connecting mem-models with classical theories," Nonlinear dynamics, 103(2), pp. 1321-1344.
[59] Bouc, R., "Forced vibrations of mechanical systems with hysteresis," Proc. Proc. of the Fourth Conference on Nonlinear Oscillations, Prague, 1967.
[60] Bouc, R., 1971, "A mathematical model for hysteresis," Acta Acustica united with Acustica, 24(1), pp. 16-25.
[61] Wen, Y.-K., 1976, "Method for random vibration of hysteretic systems," Journal of the Engineering Mechanics Division, 102(2), pp. 249-263.
[62] Lei, Y., Wu, Y., and Li, T., 2012, "Identification of non-linear structural parameters under limited input and output measurements," International Journal of Non-Linear Mechanics, 47(10), pp. 1141-1146.

List of Table Captions

List of Figure Captions

Table 1 Summary of tunable parameters and features of different model classes

Model	Tunable Parameters	Interpretability	Dataset size
GP-NARX	Covariance function, likelihood covariance noise, and number of lags	Easy to interpret	Computational complexity is $O(n^3)$
LSTM, CNN-LSTM, CNN-BLSTM	Number of layers, number of neurons, batch size, number of epochs, dropout ratio, activation function, weight decay ratio, and number of lags	Ongoing research topic to develop interpretable models	Can handle “big data”/very large datasets

Table 2 MSE comparison of GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM (Surrogate modeling with subsampling)

Case	Data Dimension	GP-NARX	LSTM	CNN-LSTM	CNN-BLSTM	BEST
1	(40,2)	0.15142	0.02912	0.02915	**0.02147**	0.02147
2	(70,2)	**0.00620**	0.01923	0.01288	0.01113	0.00620
3	(66,2)	0.02138	0.01661	0.01479	**0.01278**	0.01278
4	(80,2)	0.09544	0.01809	0.01409	**0.01303**	0.01303
5	(30,2)	0.01953	0.02891	0.01390	**0.01288**	0.01288
6	(190,2)	**0.00344**	0.01414	0.01486	0.01359	0.00344
7	(150,2)	0.01348	**0.00885**	0.01007	0.00923	0.00885
8	(200,2)	0.01510	**0.01208**	0.01358	0.01309	0.01208
9	(150,2)	**0.00694**	0.01402	0.01628	0.01513	0.00694
10	(300,2)	**0.00321**	0.01367	0.01317	0.01270	0.00321
11	(50,2)	0.01329	**0.01227**	0.01510	0.01516	0.01227
12	(30,2)	**0.00588**	0.01020	0.01003	0.00781	0.00588
13	(45,2)	0.24415	0.04063	0.03120	**0.01823**	0.01823
14	(60,2)	**0.00668**	0.01395	0.01108	0.01012	0.00668
Average	(104,2)	0.04330	0.01798	0.01573	**0.01331**	0.01331

Table 3 MSE comparison of GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM (Surrogate modeling using all available data without subsampling)

Case	Data Dimension	GP-NARX	LSTM	CNN-LSTM	CNN-BLSTM	BEST
1	(40,2)	0.0354	**0.0094**	0.0162	0.0110	0.0094
2	(70,2)	0.0200	0.0096	0.0130	**0.0092**	0.0092
3	(66,2)	0.0222	**0.0112**	0.0128	0.0118	0.0112
4	(80,2)	0.0231	0.0115	0.0145	**0.0100**	0.0100
5	(30,2)	**0.0071**	0.0104	0.0120	0.0120	0.0071
6	(190,2)	**0.0116**	0.0127	0.0145	0.0120	0.0116
7	(150,2)	**0.0106**	0.0139	0.0156	0.0125	0.0106
8	(200,2)	**0.0038**	0.0145	0.0136	0.0122	0.0038
9	(150,2)	0.0123	0.0118	0.0141	**0.0111**	0.0111
10	(300,2)	**0.0076**	0.0118	0.0144	0.0103	0.0076
11	(50,2)	**0.0047**	0.0104	0.0144	0.0090	0.0047
12	(30,2)	0.0520	0.0156	0.0197	**0.0140**	0.0140
13	(45,2)	0.0870	0.0105	0.0104	**0.0086**	0.0086
14	(60,2)	0.0273	0.0083	0.0125	**0.0081**	0.0081
Average	(104,2)	0.0232	0.0115	0.0141	**0.0108**	0.0108

Table 4 Comparison of GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM for Example 2

Case	Data Dimension	*MSE* (×10^{-8})				
		GP-NARX*	LSTM	CNN-LSTM	CNN-BLSTM	BEST
1	(3000,9,3)	**1.29**	28.3	9.59	18.3	1.29
2	(2500,9,3)	**0.31**	22.5	5.02	12.5	0.31
3	(2700,9,3)	**3.44**	31.8	10.6	32.6	3.44
4	(2000,9,3)	**2.27**	45.8	10.4	32.3	2.27
5	(3100,9,3)	**2.75**	42.7	10.4	36.3	2.75
6	(1800,9,3)	**2.69**	44.5	10.1	32.2	2.69
7	(1900,9,3)	**2.68**	44.9	11.5	32.1	2.68
8	(2100,9,3)	**2.93**	42.5	10.1	34.7	2.93
Average	(2750,9,3)	**2.29**	37.9	9.7	28.9	2.29

** Note that GP-NARX is trained based on subsampling.*

Table 5 MSE comparison of the four surrogate modeling methods for Drift of DOF1

Case	Data Dimension	GP-NARX	LSTM	CNN-LSTM	CNN-BLSTM	BEST
1	(520,2)	0.025	**0.015**	0.178	0.160	0.015
2	(455,2)	**0.022**	0.023	0.079	0.085	0.022
3	(470,2)	**0.007**	0.027	0.116	0.124	0.007
4	(438,2)	**0.006**	0.021	0.074	0.107	0.006
5	(289,2)	**0.009**	0.010	0.070	0.071	0.009
6	(407,2)	**0.005**	0.013	0.085	0.103	0.005
7	(518,2)	**0.006**	0.013	0.089	0.110	0.006
8	(393,2)	**0.008**	0.029	0.060	0.049	0.008
Average	(436,2)	**0.011**	0.019	0.094	0.101	0.011

Table 6 MSE comparison of the four surrogate modeling methods for Drift of DOF2

Case	Data Dimension	GP-NARX	LSTM	CNN-LSTM	CNN-BLSTM	BEST
1	(520,2)	0.031	**0.013**	0.110	0.169	0.013
2	(455,2)	**0.034**	0.060	0.259	0.207	0.034
3	(470,2)	**0.016**	0.084	0.151	0.168	0.016
4	(438,2)	**0.012**	0.067	0.121	0.130	0.012
5	(289,2)	**0.014**	0.020	0.209	0.194	0.014
6	(407,2)	**0.010**	0.012	0.073	0.108	0.010
7	(518,2)	**0.012**	0.022	0.124	0.120	0.012
8	(393,2)	**0.006**	0.023	0.075	0.228	0.006
Average	(436,2)	**0.017**	0.038	0.140	0.166	0.017

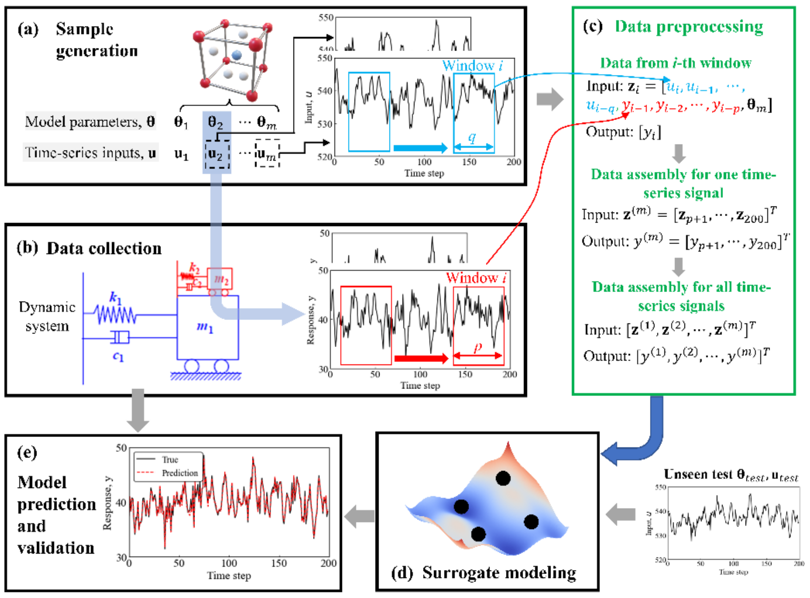

Fig. 1 Overview of surrogate modeling for nonlinear dynamic systems

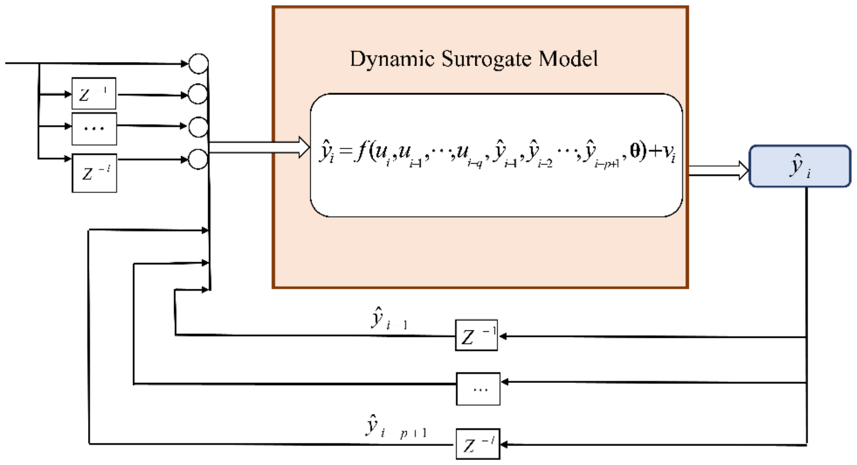
Dynamic Surrogate Model
$\hat{y}_i = f(u_i, u_{i-1}, \cdots, u_{i-q}, \hat{y}_{i-1}, \hat{y}_{i-2}, \cdots, \hat{y}_{i-p+1}, \theta) + v_i$
$\hat{y}_i$

Fig. 2 Schematic of recursive single-step prediction for a univariate dynamic predictive model

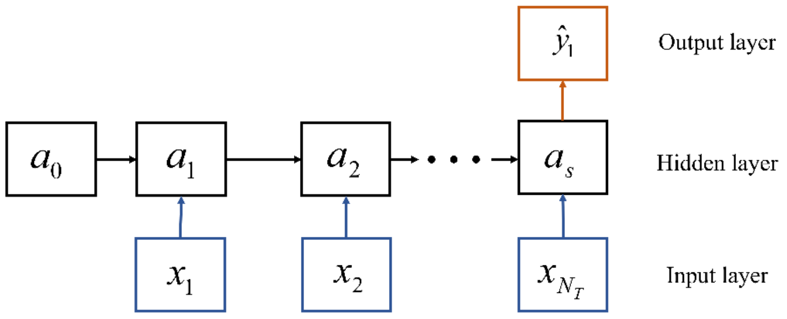

Fig. 3 Many-to-one recurrent neural network

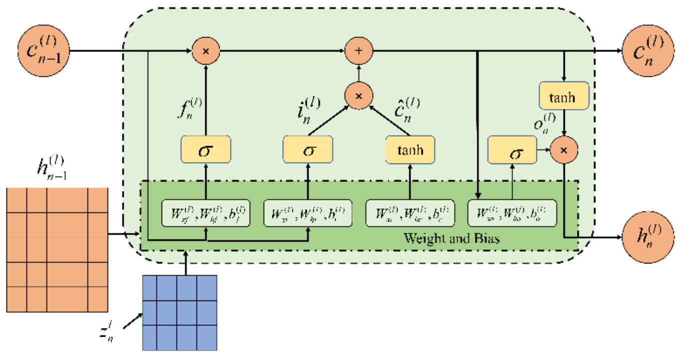

Fig. 4 Diagram of an LSTM cell

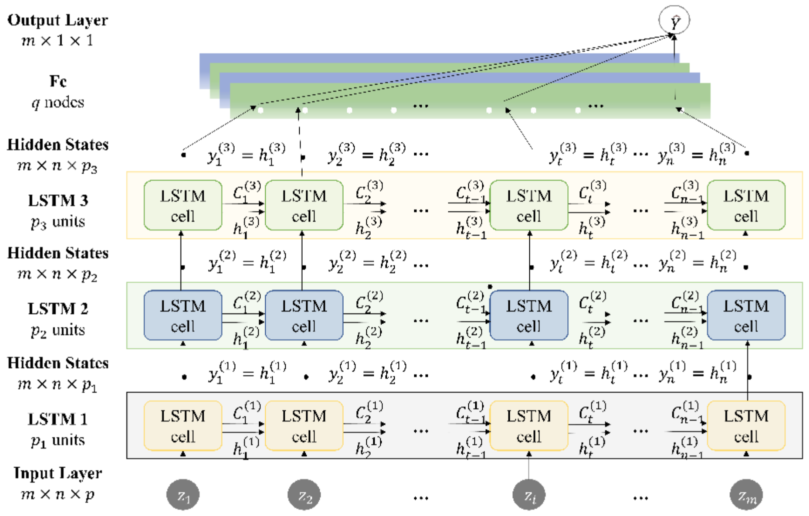

Fig. 5 A three-layers LSTM architecture

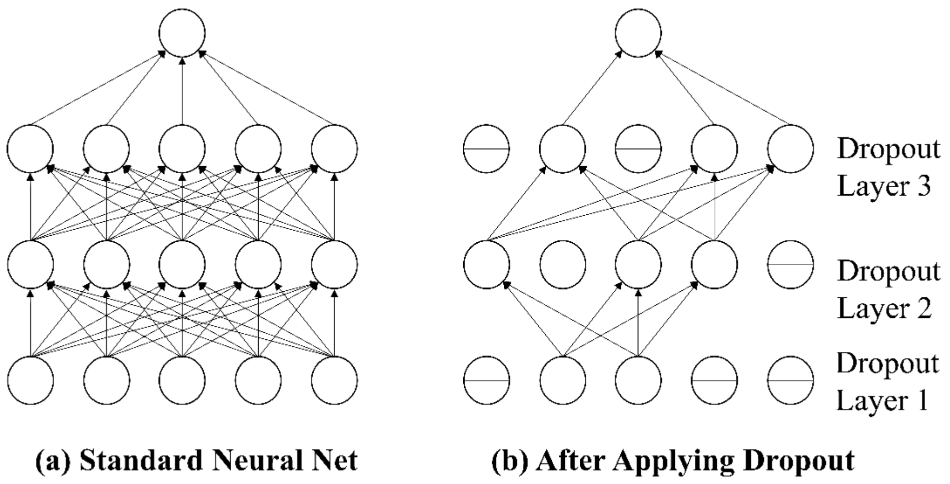

Fig. 6 Illustration of dropout in LSTM

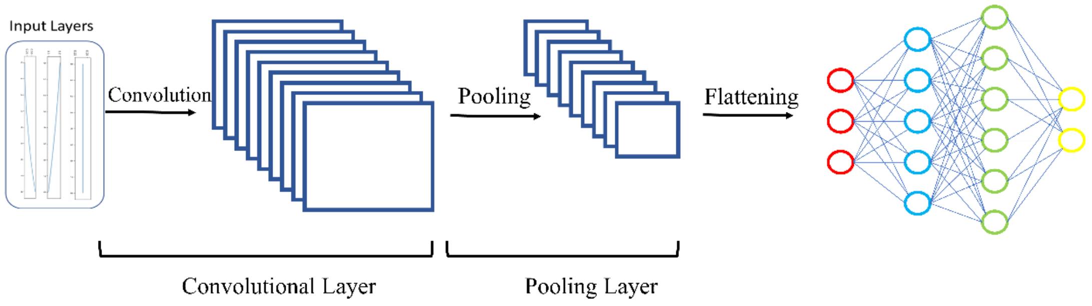

Fig. 7 ConvNet Architecture

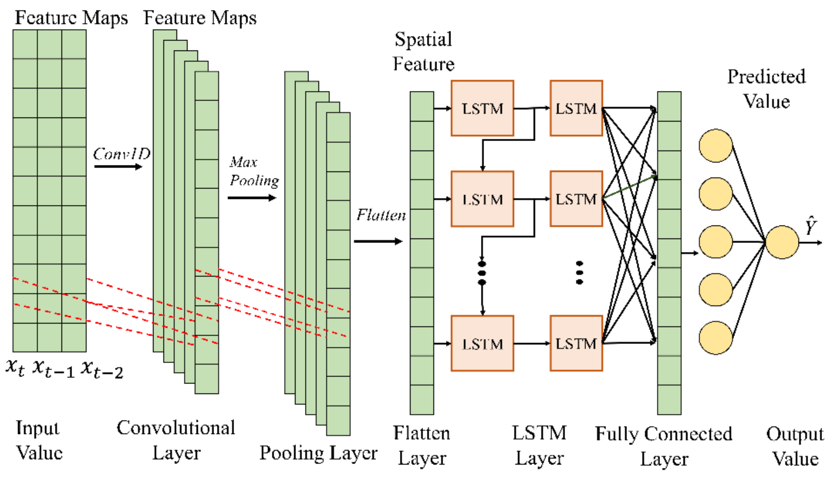

Fig. 8 CNN-LSTM architecture

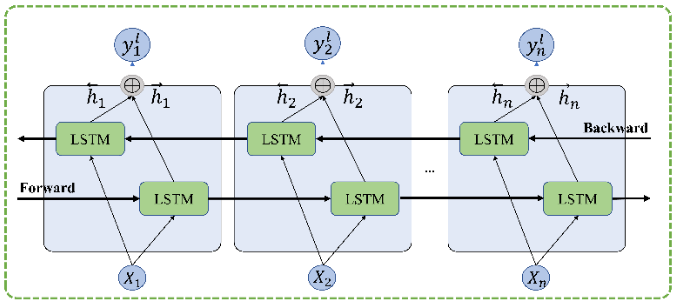

Fig. 9 Architecture of bidirectional LSTM

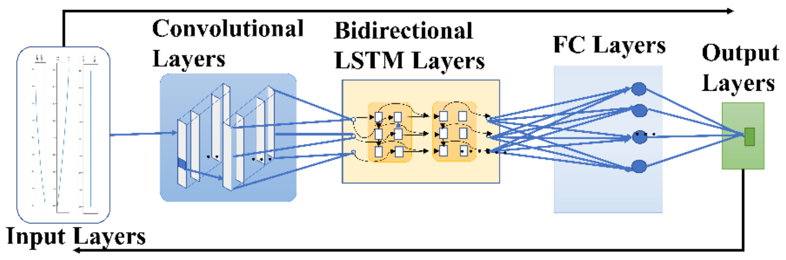

Fig. 10 Illustration of CNN-BLSTM architecture

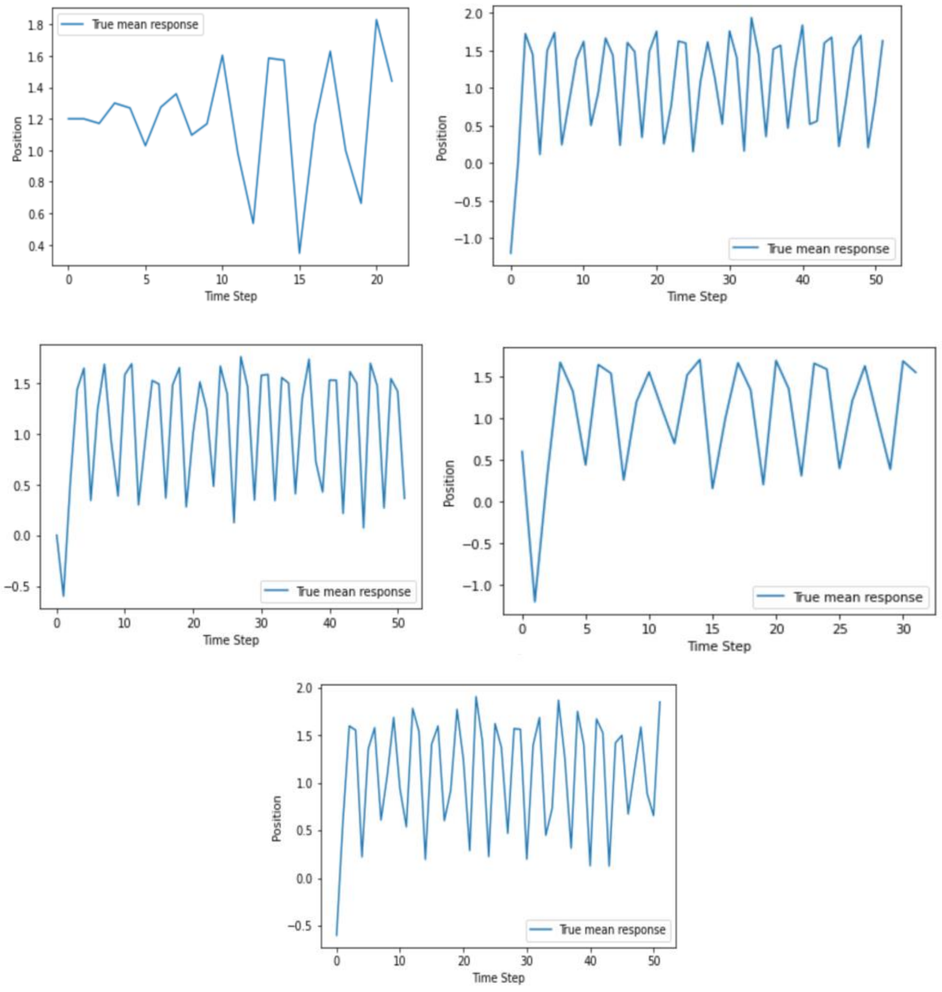

Fig. 11 Five time-series training data

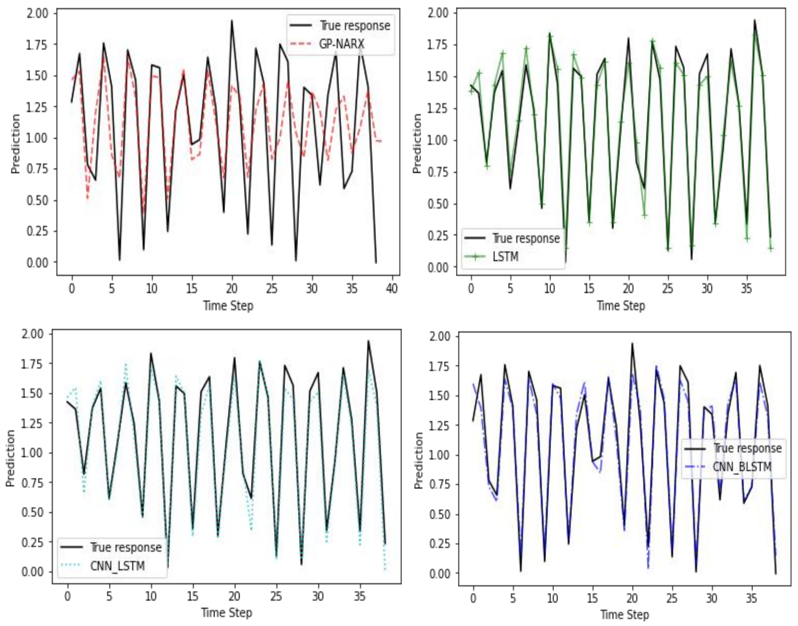

Fig. 12 Comparison of prediction and true response for Case 1 for GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM

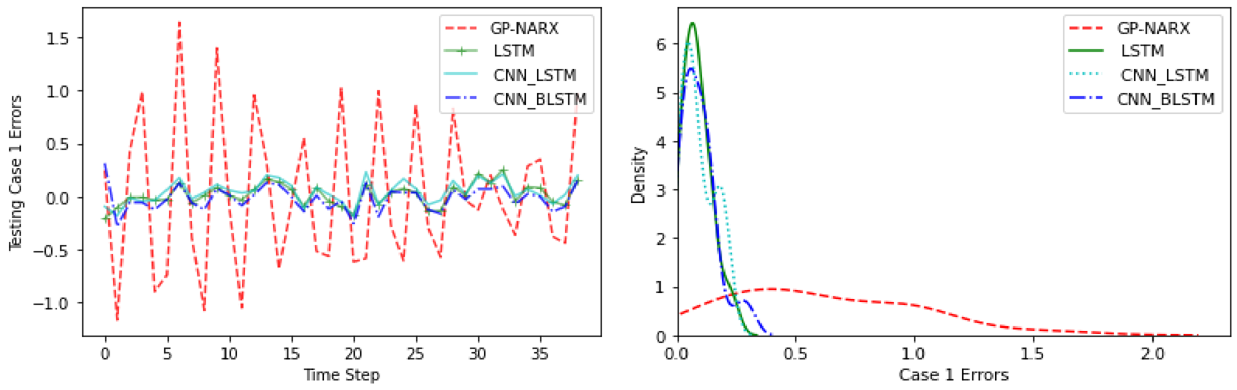

Fig. 13 Comparison of prediction errors for the case given in Fig.12

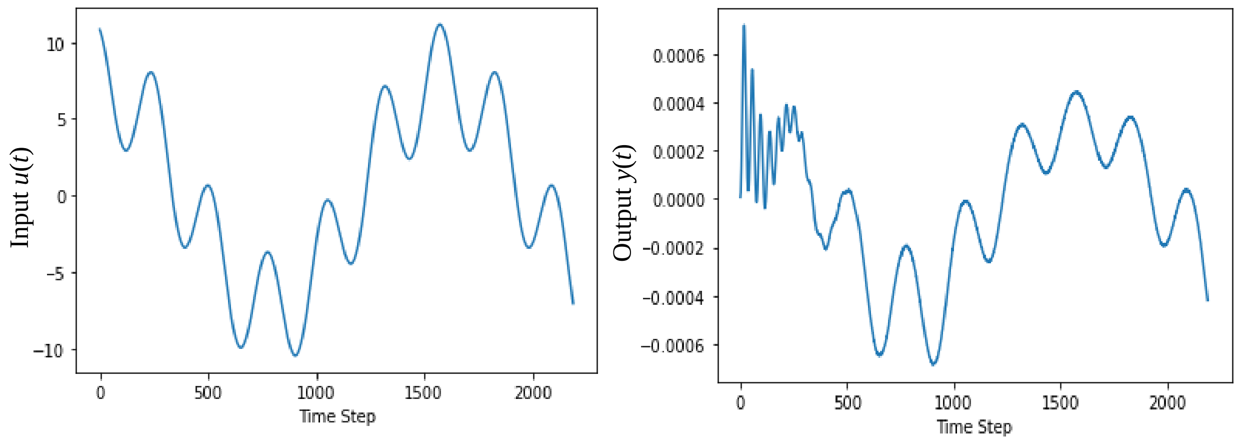

Fig. 14 An example of training excitation and corresponding dynamic response

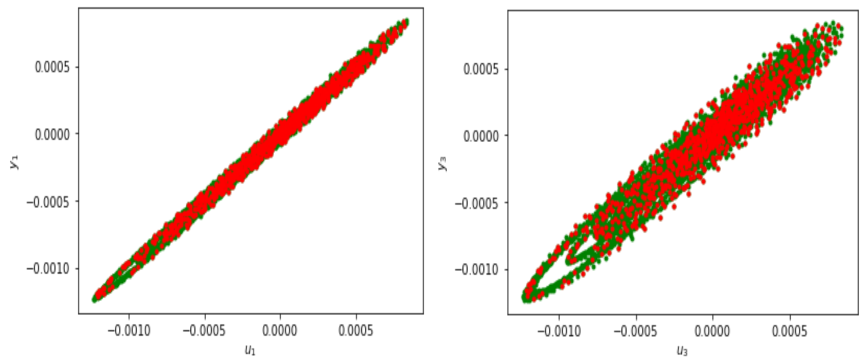

Fig. 15 A comparison of sub-selected training data and the original training data

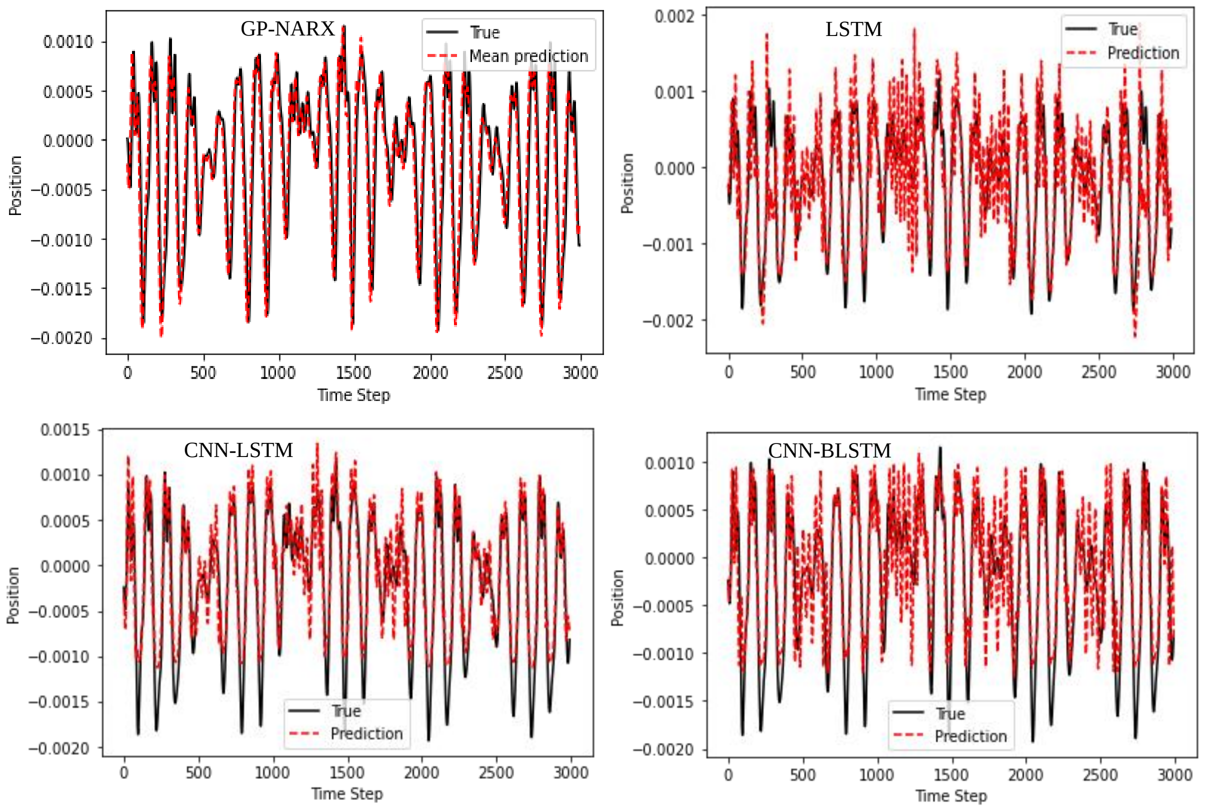

Fig. 16 Comparison of prediction and true response for Case 1 for GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM

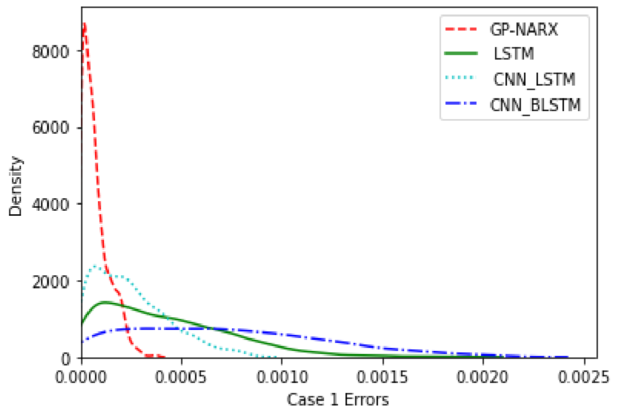

Fig. 17 Comparison of prediction error distribution for a testing case

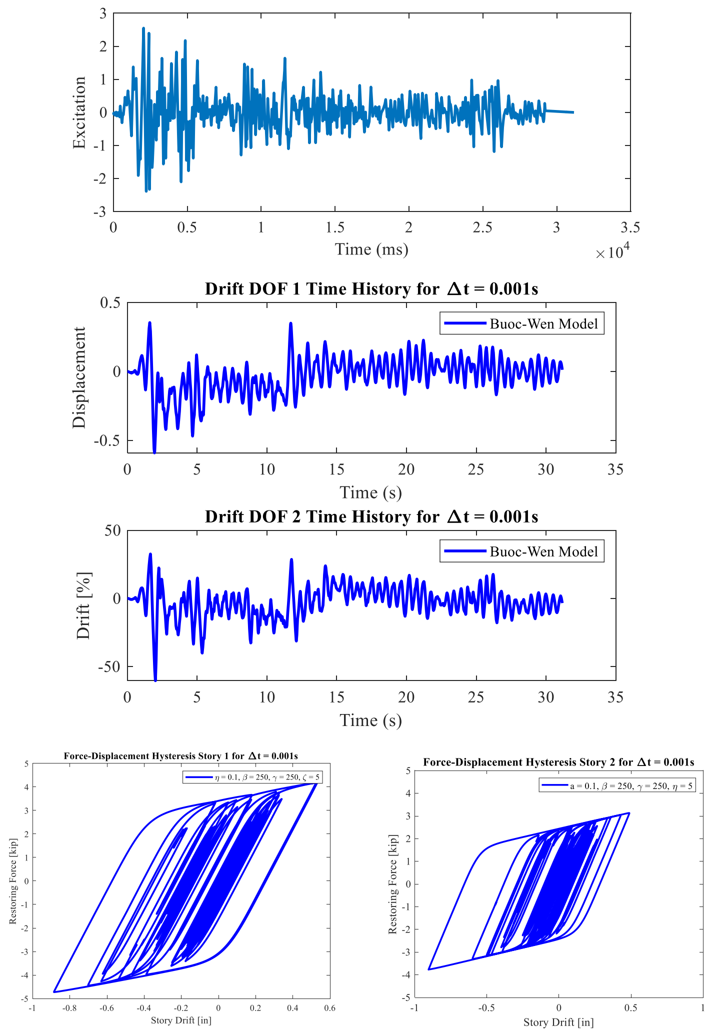

Fig. 18 An example of random excitation and corresponding dynamic responses

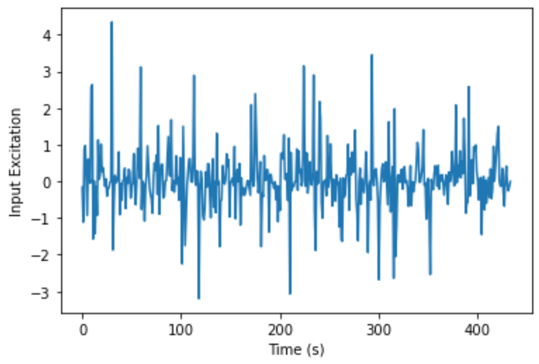

Fig. 19 Comparison of prediction and true response of a testing case for GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM

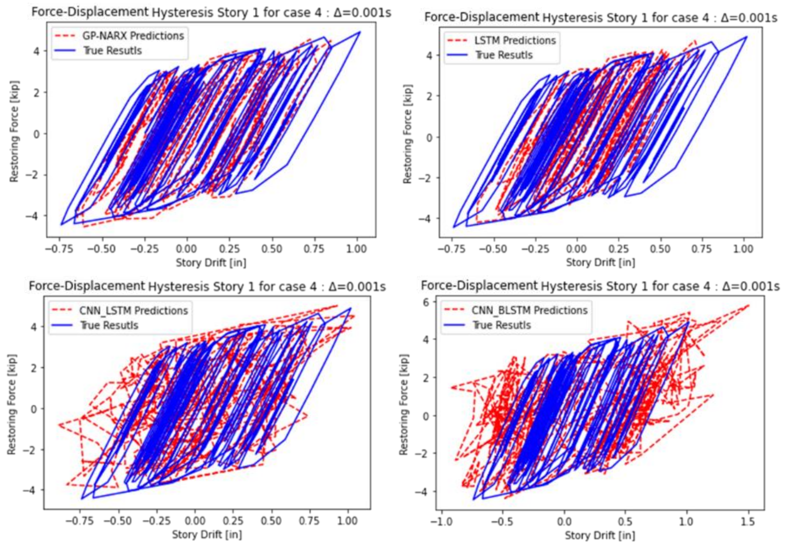

Fig. 20 Comparison of prediction and true response of a testing case for GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM (force-displacement hysteresis of story 1)

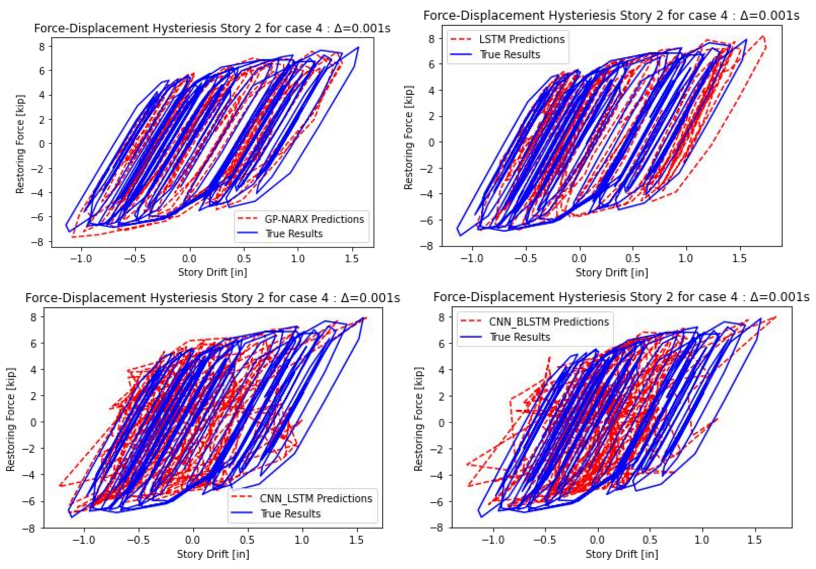

Fig. 21 Comparison of prediction and true response of a testing case for GP-NARX, LSTM, CNN-LSTM, and CNN-BLSTM (force-displacement hysteresis of story 2)

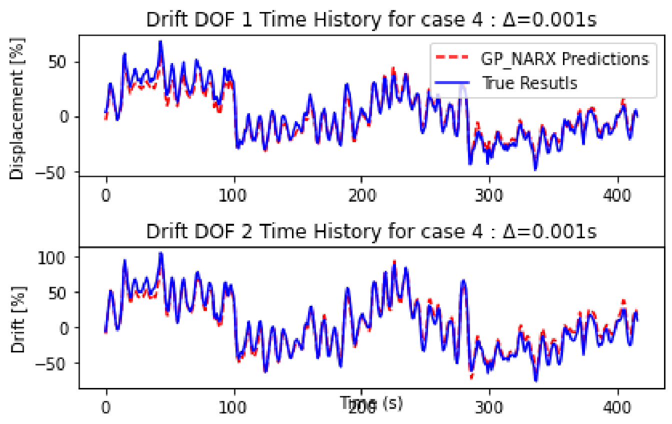

Fig. 22 Comparison of prediction and true response of drifts for GP-NARX

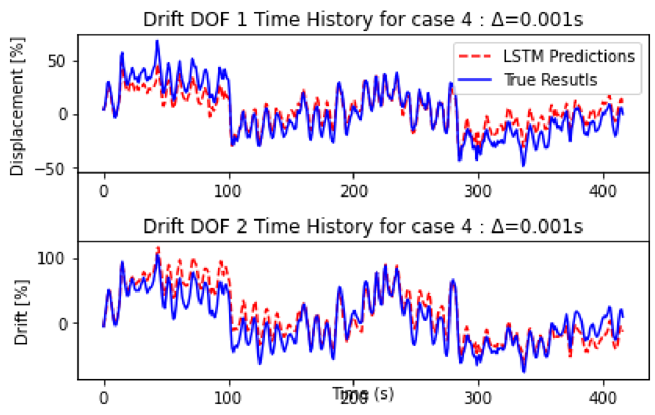

Fig. 23 Comparison of prediction and true response of drifts for LSTM

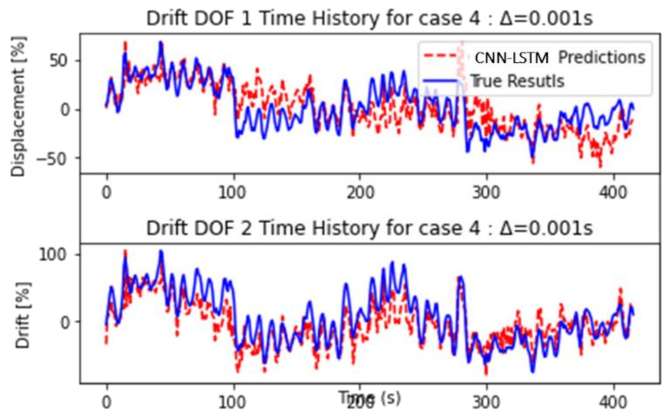

Fig. 24 Comparison of prediction and true response of drifts for CNN-LSTM

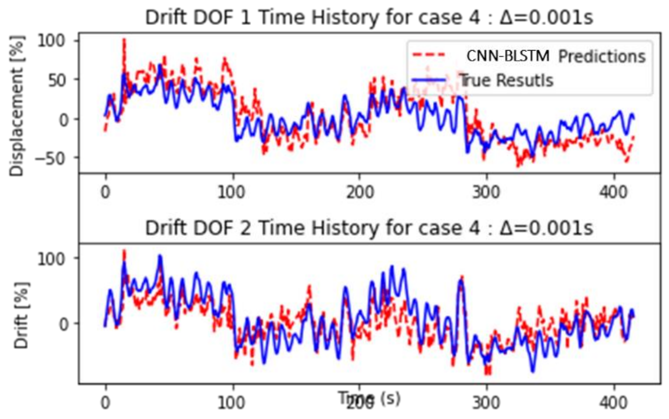

Fig. 25 Comparison of prediction and true response of drifts for CNN-BLSTM

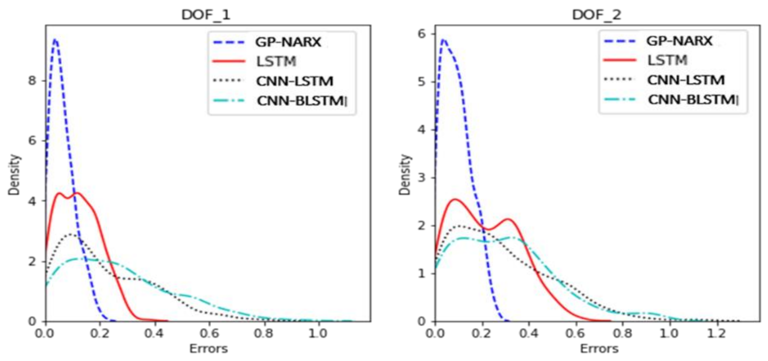

Fig.26 Comparison of probability density functions of prediction errors for a testing case