Original Article

Institution of MECHANICAL ENGINEERS
Journal of AEROSPACE ENGINEERING

Proc IMechE Part G:
J Aerospace Engineering
2023, Vol. 237(7) 1499–1510
© IMechE 2022
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/09544100221128998
journals.sagepub.com/home/pig
Sage

Updating multi-fidelity structural dynamic models for flexible wings with feed-forward neural network

Yanxin Huang and Weihua Su

Abstract
In multidisciplinary design optimization of aerospace structures (e.g., a flexible wing), it may be convenient and practical to break such a complex problem into multi-fidelity, multi-stage design problems. Structural model updating is needed in multi-fidelity, multi-stage optimizations to ensure the consistency of models with different fidelity. However, due to the inequality in structural parameters, there exists a fundamental difficulty in the model updating from a lower fidelity model to a higher fidelity model. In this paper, a feed-forward neural network is applied to determine the structural dynamic characteristics of a higher fidelity model based upon a lower fidelity model. The feasibility of this approach is demonstrated by updating beam-like wings to a thin shell-based model and a one-cell wing box model, respectively. The quality and accuracy of model updating using the proposed method are also discussed regarding the neural network structure and sample size.

Keywords
model updating, artificial neural network, aerospace structure, flexible wings, multi-fidelity models

Date received: 29 July 2021; accepted: 6 September 2022

Introduction

In recent years, multi-fidelity, multi-stage approaches have been proposed for the multidisciplinary design of aerospace structures. These approaches leverage structure models of multiple fidelity. For example, beam models can be used in low fidelity analysis providing a fast aeroelastic and performance design. In contrast, shell and wing box models are used in the high-fidelity analysis for detailed structural stress distribution. As the design optimization may be conducted parallelly with the models at different levels of fidelity, an exchange of model properties is needed for the lower-to-higher and higher-to-lower updating. These updating processes ensure that the parallel optimizations are pertinent to the same wing configuration. Therefore, the structure designed in one fidelity satisfying its constraints can provide a reference to the other fidelity for another design process and problem. The design cycle is sped up as long as the designed structure on one fidelity level and the reference on the other fidelity level represent the same wing configuration.

There have been previous attempts for structural model updating from higher fidelity models to lower fidelity models, such as the variational asymptotic method (VAM) [1,2] and the variational asymptotic beam section analysis (VABS).[3–6] Based on these ideas, a three-dimensional elasticity problem for a slender structure can be split into a two-dimensional cross-sectional analysis and a one-dimensional beam analysis. Thus, the updating or dimensional reduction from higher-fidelity to lower-fidelity structures can be conducted with fewer restrictions. Note that the term “fidelity” is used synonymously with “dimension” for structural models in this paper.

However, it is still a novel field for structural model updating from lower fidelity models to higher fidelity models, which is essentially a dimensional expansion problem. The artificial neural network (ANN) provides an alternative to traditional direct and iterative methods due to its powerful training process. One of the appealing features of ANN is that it can approximate the nonlinear mapping of input and output data without knowing the relationship between them. All input and output data of ANN are physically meaningful. Furthermore, ANN does not require any derivative, making the application more convenient. The quality of ANN depends on discrete training samples. Although the requirement of generating

Department of Aerospace Engineering and Mechanics, The University of Alabama, Tuscaloosa, AL, USA

Corresponding author:
Weihua Su, Department of Aerospace Engineering and Mechanics, The University of Alabama, Tuscaloosa, AL 35487-0280, USA.
Email: suw@eng.ua.edu

the samples may offset the convenience of ANN, the approaches with ANN are still efficient if the samples already exist in the database.

Artificial neural networks have been applied for structural monitoring and damage detection in a variety of engineering fields, including beams,[7] bridges,[8] building frames,[9] pipelines,[10] and helicopter airframes.[11] Deep learning neural networks,[12] elaborated neural networks with the Bayesian statistical framework,[13] and the Kriging interpolation[14] have also been studied. In this paper, a multi-layer feed-forward neural network is applied since its generality for nonlinear mappings with two hidden layers has been confirmed.[15] The capability of pattern matching makes the multi-layer feed-forward neural network a promising tool for updating or dimensional expansion problems. In addition, the number of hidden neurons and sample size have been recognized to significantly impact the performance of neural networks. An insufficient number of hidden neurons or sample size may result in a poor neural network, whereas too many hidden neurons or sample size may cause overfitting. Recently, the Local Linear Model Tree has been developed to determine the number of neurons for neural-fuzzy networks.[16] However, the model tree optimization may make the feed-forward neural network far more complex and offset its convenience. Hence, there is no strict rule to determine these numbers for feed-forward neural networks by far.

This paper focuses on studies of updating cantilever beams to higher-fidelity shell and wing box configurations. The feed-forward ANN with two hidden layers is implemented to transfer modal characteristics and total mass to higher fidelity models for pattern matching. Thus, linear static and modal analyses in the multi-fidelity, multi-stage approach are guaranteed to be consistent. The proposed neural network is intended to simultaneously determine the higher-fidelity model's geometric parameters and material properties. Section 2 is devoted to the fundamentals of neural networks and error comparisons between the predicted and measured models. Section 3 deals with two updating cases. One is the updating from a beam-like wing to a thin shell-like wing, and the other is the updating from a beam-like wing to a one-cell wing box. Trial studies are also carried out for the proper sample size and the number of hidden neurons in each case. Section 4 concludes the implementation of neural networks for model updating and discusses future research directions.

Fundamentals of model updating using neural networks

The flowchart of the proposed model updating is shown in Figure 1. The original beam properties are fed into the modal analysis to obtain the natural frequencies and mode shape vectors of the desired modes. All these modal characteristics and the total mass of the original beam are stored in the column vector $\boldsymbol{X}_{low}$. The design center of the higher-fidelity model can impact the quality and computational cost of the neural network. It can be approximately determined via the moment of inertia of a specific cross-section. A well-selected center can lead to an updating process with a smaller training space. On the contrary, an arbitrarily picked center requires a larger training space, a larger training population, and much more computational effort. In this study, the inverse process of the VABS approach[17] is used for this approximation.

Higher-fidelity samples around the design center within a design space are fed into the MSC.Nastran for the modal analysis. Modal characteristics of each sample are obtained and stored in vector $\boldsymbol{X}_{high}$. All sampled vectors and $\boldsymbol{X}_{low}$ are fed into the feed-forward neural network. The neural network compares $\boldsymbol{X}_{high}$ and $\boldsymbol{X}_{low}$ for an error vector. Thus, the paired input-output of the neural network is the higher-fidelity design and the corresponding error vector. The neural network aims to figure out the higher-fidelity design with the minimum error vector via the training, cross-validation, refinement, and inverse design process.

In a multi-fidelity, multi-stage aeroelastic design, the original beam properties for updating represent the beam-like wing with optimal overall performance and aerodynamic shape. They are usually obtained via a lower-fidelity aeroelastic optimization. However, this study focuses on demonstrating the feasibility of this updating approach. Therefore, the beam properties are given without the aeroelastic optimization process.

Comparison of modal characteristics

The modal characteristics, including natural frequencies and mode shape vectors, are adopted since they can reflect the inherent structural dynamic properties irrespective of excitations.[7,9] Thus, $\boldsymbol{X}_{low}$ and $\boldsymbol{X}_{high}$ are expressed as

$$\begin{aligned}\boldsymbol{X}_{low} &= \left\{f_{low\,1},\ldots,f_{low\,N_m},\boldsymbol{\varphi}_{low\,1},\ldots,\boldsymbol{\varphi}_{low\,N_m},\ldots,m_{low}\right\}\\ \boldsymbol{X}_{high} &= \left\{f_{high\,1},\ldots,f_{high\,N_m},\boldsymbol{\varphi}_{high\,1},\ldots,\boldsymbol{\varphi}_{high\,N_m},\ldots,m_{high}\right\}\end{aligned} \tag{1}$$

where the superscripts "*low*" and "*high*" represent the fidelity of each model. f represents the natural frequency. $\boldsymbol{\varphi}$ represents the mode shape vector. The subscript "N_m" represents the number of selected modes in the comparison. As previously discussed, an error vector $\boldsymbol{\varepsilon}$ between $\boldsymbol{X}_{low}$ and $\boldsymbol{X}_{high}$ is required for the comparison. $\boldsymbol{\varepsilon}$ is constructed as

$$\boldsymbol{\varepsilon} = \left\{\varepsilon_{f\,1},\ldots,\varepsilon_{f\,N_m},\varepsilon_{\varphi 1},\ldots,\varepsilon_{\varphi N_m},\ldots,\varepsilon_m\right\} \tag{2}$$

where the components ε_f and ε_m represent the relative error of the natural frequency and mass, respectively. ε_φ represents the root mean square error (RMSE) of the mode shape vector. With the error vector, the model updating is

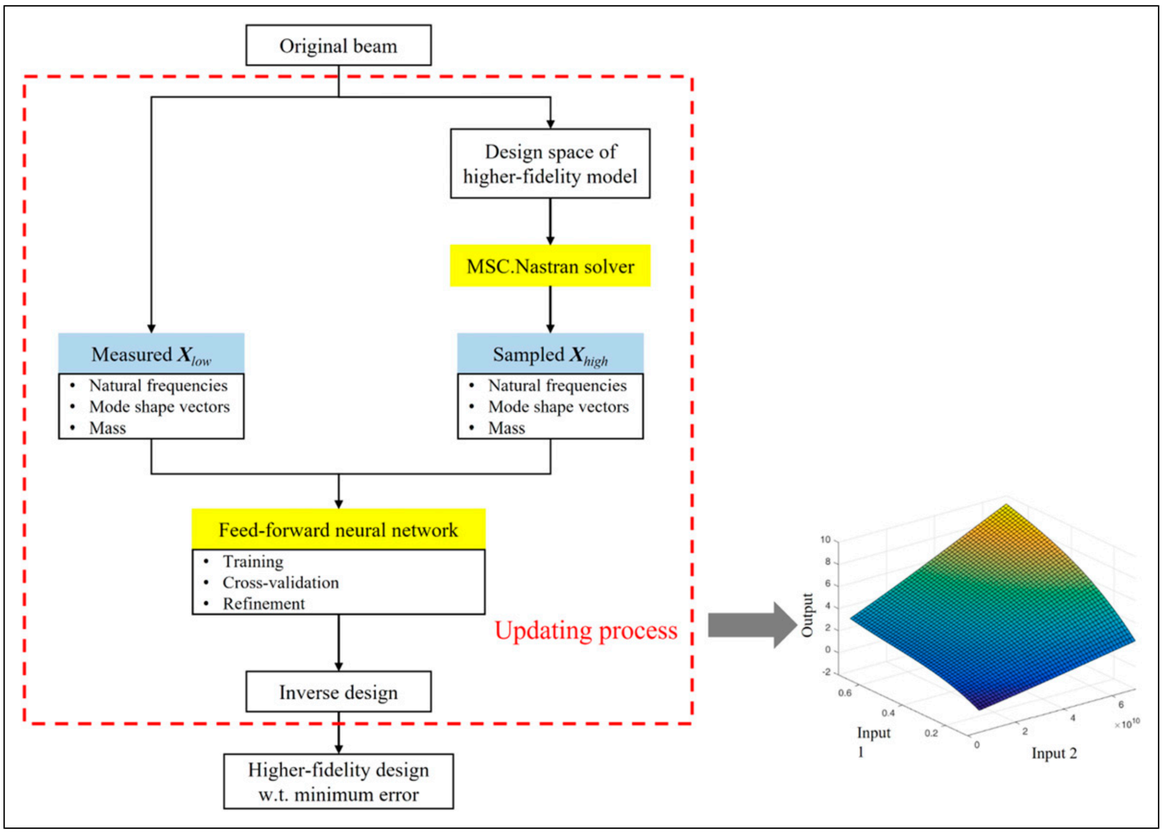

Figure 1. Flowchart of model updating.

achieved by minimizing the sum of all individual error components while satisfying all individual constraints, given as

$$\min_{X_{high}} \sum \varepsilon \quad \text{s.t.} \begin{cases} \varepsilon_f \le \varepsilon_{f,\lim} \\ \varepsilon_\varphi \le \varepsilon_{\varphi,\lim} \\ \varepsilon_m \le \varepsilon_{m,\lim} \end{cases} \tag{3}$$

Feed-forward neural network

The feed-forward neural network possesses a layered structure and allows connections from neurons in one layer to those in the forward direction. Various interconnected artificial neurons are involved in such neural networks, which can model biological neurons in the simplest form while performing complex tasks. The process of transferring an N_I-dimensional input vector to an N_O-dimensional output vector within two hidden layers is illustrated in Figure 2. This special architecture is mainly designed for approximating an unknown nonlinear mapping.

In the hidden layer, each neuron receives multiple inputs. Through activation functions, weight matrices **w**, and bias vectors ***b***, one output a is produced and transferred to the next layer in each neuron. Thus, output vectors $\boldsymbol{a}^1$ and $\boldsymbol{a}^2$ are expressed as

$$\begin{aligned} \boldsymbol{a}^1 &= \mathrm{f}^1\left(\mathbf{w}^1\mathbf{I} + \mathbf{b}^1\right) \\ \boldsymbol{a}^2 &= \mathrm{f}^2\left(\mathbf{w}^2\mathbf{a}^1 + \mathbf{b}^2\right) \\ &= \mathrm{f}^2\left(\mathbf{w}^2\mathrm{f}^1\left(\mathbf{w}^1\mathbf{I} + \mathbf{b}^1\right) + \mathbf{b}^2\right) \end{aligned} \tag{4}$$

where f^1 and f^2 usually represent the hyperbolic tangent function and linear function, respectively.[18] **I** represents the input vector. Equation (4) can be expanded as

$$\begin{aligned} a_1^1 &= \mathrm{f}^1\left(w_{1,1}^1 I_1 + w_{1,2}^1 I_2 + \cdots + w_{1,N_I}^1 I_{N_I} + b_1^1\right) \\ a_2^1 &= \mathrm{f}^1\left(w_{2,1}^1 I_1 + w_{2,2}^1 I_2 + \cdots + w_{2,N_I}^1 I_{N_I} + b_2^1\right) \\ &\vdots \\ a_{N_h}^1 &= \mathrm{f}^1\left(w_{N_h,1}^1 I_1 + w_{N_h,2}^1 I_2 + \cdots + w_{N_h,N_I}^1 I_{N_I} + b_{N_h}^1\right) \end{aligned} \tag{5}$$

and

$$\begin{aligned} a_1^2 &= w_{1,1}^2 a_1^1 + w_{1,2}^2 a_2^1 + \cdots + w_{1,N_h}^2 a_{N_h}^1 + b_1^2 \\ a_2^2 &= w_{2,1}^2 a_1^1 + w_{2,2}^2 a_2^1 + \cdots + w_{2,N_h}^2 a_{N_h}^1 + b_2^2 \\ &\vdots \\ a_{N_O}^2 &= w_{N_O,1}^2 a_1^1 + w_{N_O,2}^2 a_2^1 + \cdots + w_{N_O,N_h}^2 a_{N_h}^1 + b_{N_O}^2 \end{aligned} \tag{6}$$

where N_h represents the number of neurons in the first hidden layer. $\boldsymbol{a}^2$ represents one predicted error vector between the generated higher-fidelity model and the original lower-fidelity model. It is the final output of the neural network. In addition to the predicted error vector, each sampled higher-fidelity model is compared with the

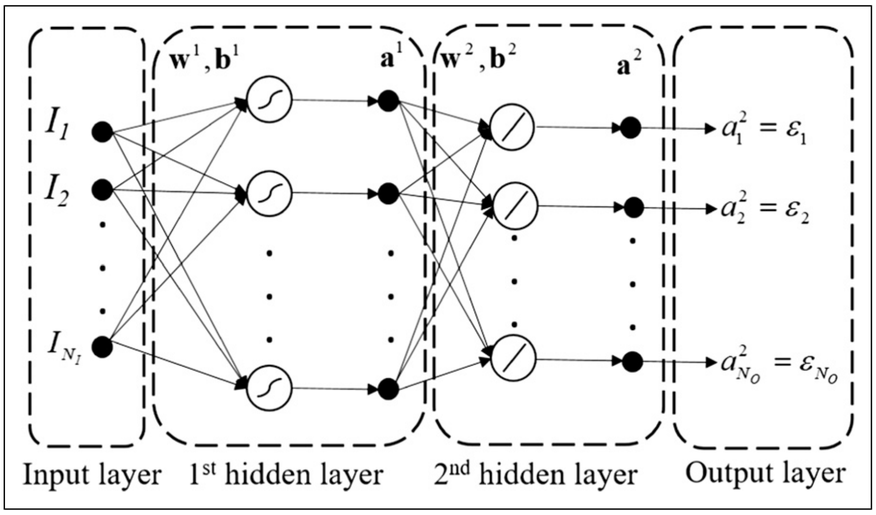

Figure 2. A typical feed-forward ANN with two hidden layers.

original lower-fidelity model, providing a measured error vector between $\boldsymbol{X}_{low}$ and $\boldsymbol{X}_{high}$. Depending on the complexity of the feed-forward neural network, all predicted error vectors obtained by neural networks should accurately match their corresponding measured error vectors.

Design of ANN

The design of an ANN generally consists of the normalization of input, the training process, the determination of neural network structure and sample size, and the refinement, if necessary.

Normalization. For the accuracy of the neural network, a normalization of input data is required since the raw training data can vary significantly in their orders (e.g., geometrical parameters vs material properties in this study). A significant difference in magnitude orders of these data may lead to an ill-conditioned neural network, making the learning process biased or inaccurate. Previous discussions[19] indicate that properly scaling the raw input and output data can avoid this problem. This is also why the error vectors are used in the comparison.

In this study, a Latin Hypercube Sampling (LHS) strategy is used to normalize all input components within a prescribed range [0,1] around the design center. The LHS divides each dimension of the input vector into equally probable intervals and allows only one point within each interval. It then randomly selects points and combines them for a normalized vector. The main advantage of LHS is its capability to cover both small and large design space problems of clustering. In addition, if some dimensions have to be dropped, the existing data are still LHS data and can be reused without reducing the number of sampling data. Thus, all training, testing, and validation data used in the following sections comprise normalized input and output error vectors.

Cross-validation and training. Once the two-layer neural network is selected, and the samples are generated, the next decision is on the neural network's structure, that is, the number of hidden neurons in each layer. As the number of hidden neurons in the second layer is determined by the output vector, only a desired number in the first hidden layer needs to be decided.

In addition, the sample size should be determined. It has been discussed as "the curse of dimensionality" by Bishop.[20] As the dimension of the input vector increases, the number of sampled vectors required for a good neural network also increases, often exponentially. However, there is no clearly defined boundary at which the sample size is manageable.

In this work, trial studies based on the k-fold cross-validation are required. As illustrated in Figure 3, the whole data set is first split into training and testing groups, respectively. Trial studies with a specific number of hidden neurons and sample size are carried out in the training group. Given the trial of a specific design, the training group is shuffled to maintain randomness and partitioned into k equal-sized subsets. Each subset serves as a validation subset only once, with all the other subsets participating in the training process. Repeating this process yields k trained neural networks and k sets of predicted outputs. The average of errors over all the validation data sets ε_{cv} approximates the prediction capability of this specific neural network, given by

$$\varepsilon_{cv}^{(l)} = \sqrt{\frac{1}{N_{cv}} \sum_{i=1}^{N_{cv}} \sum_{j=1}^{N_O} \left(a_{i,j}^2 - t_{i,j}\right)^2}$$
$$\varepsilon_{cv} = \frac{1}{k} \sum_{i=1}^{k} \varepsilon_{cv}^{(l)} \tag{7}$$

where N_{cv} is the dimension of validation subsets, and $\boldsymbol{t}$ is the measured error between $\boldsymbol{X}_{low}$ and $\boldsymbol{X}_{high}$. Among the trial studies, the number of hidden neurons in the second layer and the sample size are usually determined with

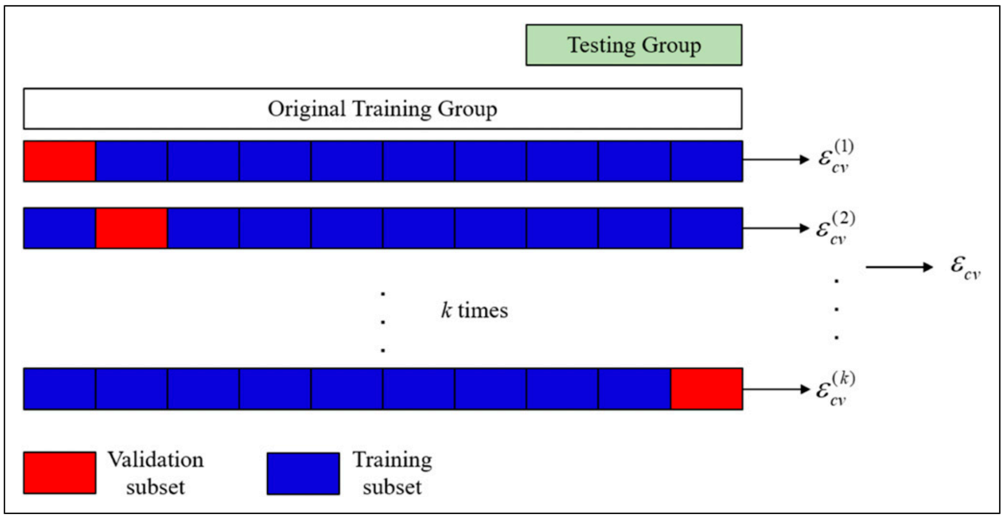

Figure 3. k-fold cross-validation.

a minimum ε_{cv}. Multiple times of random initialization can be conducted to determine variables without being trapped within a local minimum.

After the sample size and the structure are determined, all subsets in the training group (both training and validation) are used to explore the desired values of weights and biases. The neural network is well trained when all predicted outputs $\boldsymbol{a}^2$ match the measured errors $\boldsymbol{t}$ in the training group. The sum of the squared error for all pairs in the training group is obtained as the training error ε_{tr}

$$\varepsilon_{tr} = \sum_{i=1}^{N_{tr}} \left(\boldsymbol{t}_i - \boldsymbol{a}_i^2\right)^T \left(\boldsymbol{t}_i - \boldsymbol{a}_i^2\right) \tag{8}$$

where N_{tr} is the dimension of the training group.

The Levenberg–Marquardt back-propagation (LMBP) algorithm, a variation of the Steepest Descent back-propagation (SDBP), is used to train the feed-forward neural network. The weights and biases are initialized randomly and updated based on the Jacobian and Hessian matrices. The LMBP algorithm provides an efficient convergence rate. Further information about this algorithm can be found in Magar et al.[21] ε_{tr} determines the termination of the training, while the RMSE of the testing group ε_{ts} is used to assess the performance of the trained neural network.

Refinement of neural network. When the feed-forward neural network is established with small ε_{ts}, it should be able to approximate the error vector out of sampled models in the design space. The optimal higher-fidelity structural model is then obtained via the inverse design by minimizing the sum of error components in the error vector. The *fmincon* optimizer is used for this inverse design, treating the normalized design center as the initial condition of the *fmincon* optimizer.

This optimal higher-fidelity model is found based upon the neural network model. If one uses the optimum properties to create the FE model, it may not be the exact optimum solution. In this case, the neural network must be refined around the first optimal design.

The refinement procedure is shown in Figure 4. The initial optimal design is treated as the new design center. Compared to the initial design space, the refined neural network may require fewer but more clustered samples (see the green box in Figure 4). Modal analysis results of all these new samples are obtained and compared to $\boldsymbol{X}_{low}$ for error vectors. The architecture of the initial neural network and the refined neural network are the same. Therefore, the neural network for the refinement keeps the same number of hidden-layer neurons. The refinement also requires the training process and the inverse design. The refinement is terminated when the discrepancy is smaller than the tolerance.

Numerical studies

Two model updating cases are demonstrated in this section. In the first case, a beam-like wing is updated to a two-dimensional thin shell-based model. In the second one, a beam-like wing is updated to a three-dimensional one-cell wing box model. In both cases, the 10-fold cross-validation is performed to determine the number of hidden neurons and the sample size.

Model updating from beam to shell model

Objective function for shell updating. A uniform, straight beam-like wing with a large bending stiffness ratio (EI_{in}/EI_{out}) of 1000 is used to generate a thin shell wing model made up of isotropic material. The wing semi-span is a fixed constant of 13.2 m. Hence, the design parameters of the thin shell model are a combination of geometric parameters and material properties, including the width w_{shell}, thickness t_{shell}, Young's modulus E_{shell}, Poisson's ratio μ_{shell}, and density ρ_{shell} of the shell model, as shown in Figure 5.

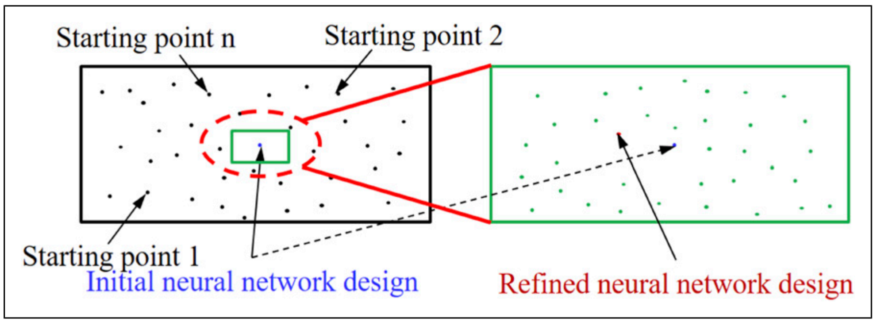

Figure 4. Refinement of ANN.

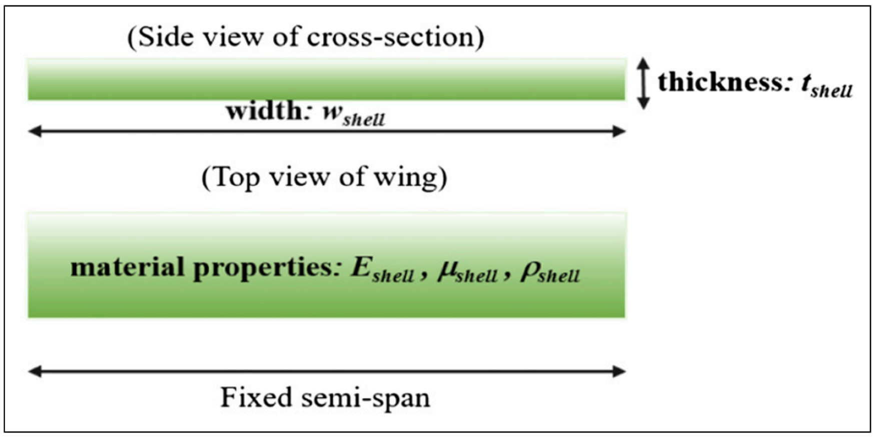
(Side view of cross-section)
thickness: t_{shell}
width: w_{shell}
(Top view of wing)
material properties: E_{shell}, μ_{shell}, ρ_{shell}
Fixed semi-span

Figure 5. Design variables of the thin shell.

Table 1. Original beam properties to generate the shell model.

Beam properties	Value
Semi-span, m	13.2
Out-of-plane bending rigidity, N·m^2	5.63 × 10^5
In-plane bending rigidity, N·m^2	6.25 × 10^8
Torsional rigidity, N·m^2	9.00 × 10^5
Mass per span, kg/m	70
Rotational moment of inertia, kg·m^2	8

By setting the tolerance of individual error as 0.05, equation (3) is formulated as

$$\min_{\mathbf{x}_{shell}} \sum \varepsilon(\boldsymbol{x}_{shell})$$
$$\boldsymbol{x}_{shell} = \{w_{shell}, t_{shell}, E_{shell}, \mu_{shell}, \rho_{shell}\}$$
$$\begin{cases} \varepsilon_{f_i} \le 0.05 & \\ \varepsilon_{\varphi_i} \le 0.05 & i = 1,2,3,4 \\ \varepsilon_m \le 0.05 & \\ \boldsymbol{x}^{lb}_{shell} \le \boldsymbol{x}_{shell} \le \boldsymbol{x}^{ub}_{shell} & \end{cases} \tag{9}$$

where i stands for the fundamental modes for comparisons. The first and second out-of-plane bending, the first torsion, and the first in-plane bending modes are compared for practicality concerns. The original beam properties and natural frequencies are given in Tables 1 and 2, respectively.

Generated thin shell model. As previously discussed, a model input file of shell structures in MSC.Nastran may consist of hundreds of CQUAD4 elements. The file is fed into the MSC.Nastran's SOL 103 for the modal analysis. The measured properties are then used to train the neural network.

Ten-fold cross-validation is used to evaluate the general prediction capability of different neural networks based on their complexity. The 10-fold cross-validation is performed since it is a good compromise between cost and accuracy.[22] Assume that twenty hidden neurons are involved in the first hidden layer. Eight neural networks with sample sizes varying from 100 to 450 are compared for the trial study. Similarly, six other neural networks are compared to determine the number of hidden neurons. The sample sizes of these six neural networks are 350, while the number of hidden neurons varies from 5 to 30. The number of testing and training groups are selected to be 20% and 80% of the data set, respectively. Variations of all the cross-validation, training, and testing errors are shown in Figure 6. It can be seen that ε_{cv}, ε_{ts}, and ε_{tr} decrease by increasing the sample size and the number of hidden neurons. As expected, all neural networks perform better on the training subsets than on the testing group and validation subsets. However, the training error is not used to estimate the neural network. The training error could drop to 0 as the overfitting occurs, while the testing error increases significantly. By observing the

Table 2. Original beam frequencies to generate the shell model.

Natural frequency	Value
1st out-of-plane bending, Hz	0.29
2nd out-of-plane bending, Hz	1.82
1st torsion, Hz	9.61
1st in-plane bending, Hz	6.36

Table 3. Design space of shell model's neural networks.

Variable	Initial shell's neural network	Refined shell's neural network
w_{shell}, m	0.5–1.5	0.8–1.2
t_{shell}, m	0.01–0.05	0.02–0.04
E_{shell}, GPa	100–350	200–300
μ_{shell}	0.25–0.35	0.28–0.32
ρ_{shell}, kg/m^3	2000–4000	2000–2800

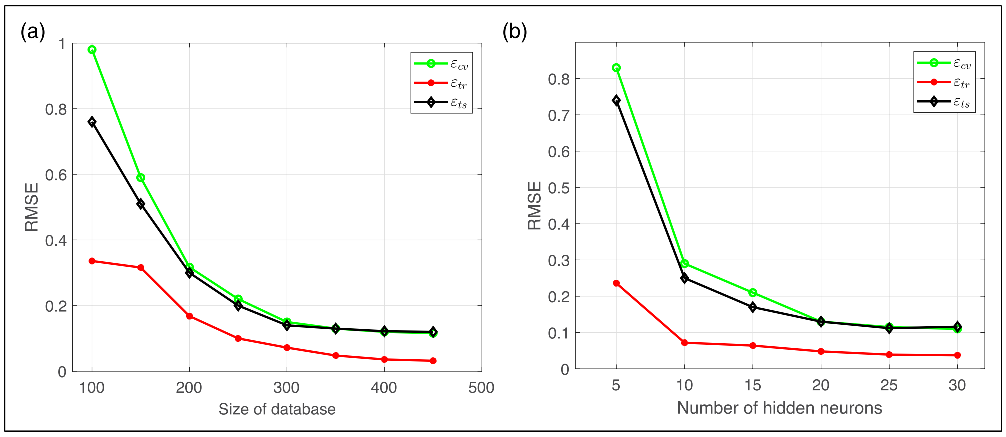

Figure 6. Performance of shell's ANN with different settings.

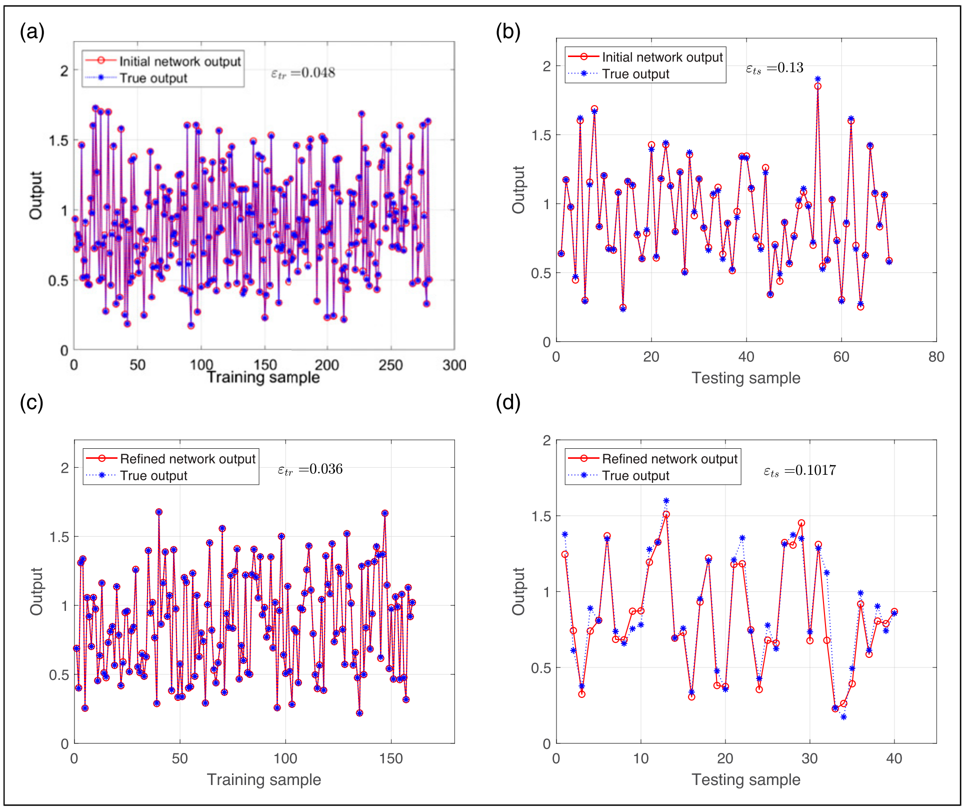

Figure 7. Comparison of shell's neural networks.

Table 4. Frequencies and mass of initially generated thin shell model.

	Value	Relative error
1st out-of-plane bending, Hz	0.30	5.05×10^{-2}
2nd out-of-plane bending, Hz	1.89	3.97×10^{-2}
1st in-plane bending, Hz	9.83	2.32×10^{-2}
1st torsion, Hz	6.27	1.42×10^{-2}
Mass, kg	898.57	2.75×10^{-2}

Table 5. Frequencies and mass of refined thin shell model.

	Value	Relative error
1st out-of-plane bending, Hz	0.29	2.70×10^{-2}
2nd out-of-plane bending, Hz	1.85	1.65×10^{-2}
1st in-plane bending, Hz	9.25	3.17×10^{-2}
1st torsion, Hz	6.22	2.21×10^{-2}
Mass, kg	921.66	2.53×10^{-3}

Table 6. Mode shape errors of generated thin shell models.

	Initial shell	Refined shell
1st out-of-plane bending	2.76×10^{-3}	2.72×10^{-3}
2nd out-of-plane bending	6.55×10^{-3}	6.54×10^{-3}
1st in-plane bending	1.90×10^{-3}	1.71×10^{-3}
1st torsion	1.75×10^{-2}	1.87×10^{-2}

Table 7. Original beam properties to generate the wing box.

Beam properties	Value
Semi-span, m	10
Out-of-plane bending rigidity, N·m^2	4.60×10^7
In-plane bending rigidity, N·m^2	2.40×10^8
Torsional rigidity, N·m^2	2.00×10^5
Mass per span, kg/m	24
Rotational moment of inertia, kg·m^2	2

Table 8. Original beam frequencies to generate the wing box.

Natural frequency	Value
1st out-of-plane bending, Hz	7.76
2nd out-of-plane bending, Hz	49.35
1st torsion, Hz	79.14
1st in-plane bending, Hz	17.73

validation and testing errors, it is found that the sample size of 350 with 20 hidden neurons can achieve excellent nonlinear mapping.

The thin shell model that matches the original beam model is generated using an initial neural network and one refined neural network. The initial neural network is designed with 350 samples (280 for training and 70 for testing) and 20 hidden neurons in the first hidden layer. The refined neural network is designed with 200 (160 for training and 40 for testing) samples and 20 hidden neurons. The training and testing results of these two neural networks are plotted in Figure 7. The design spaces of these two neural networks are shown in Table 3. It can be seen that with fewer but more clustered samples, the refined neural network provides smaller errors in both training and testing.

These trained and refined neural networks are used to determine the optimal thin shell model. The resulting frequencies, masses, and mode shape errors are shown in Tables 4–6. The thin shell model generated by the initial neural network shows minor errors except for the first out-of-plane bending mode's frequency. The error of 5.05×10^{-2} is slightly larger than the desired tolerance. The refined thin shell model mainly reduces this error to 2.70×10^{-2} while keeping all other individual errors remarkably good. Therefore, the thin shell model that matches the original beam is constructed with the geometry of w_{shell} = 0.99 m, t_{shell} = 0.032 m, and the material properties are E_{shell} = 228 GPa, μ_{shell} = 0.3, and ρ_{shell} =2237 kg/m^3.

Model updating from beam to wing box model

Objective function for wing box updating. In this section, a uniform, straight beam with a small bending stiffness ratio of 5 is used to generate a one-cell wing box with a rectangular cross-section. Tables 7 and 8 show the properties and natural frequencies of the original beam. For practicality concerns, this study does not consider the layout of internal ribs and spars.

Some assumptions are made to simplify the updating process. They are (1) the wing box is made of isotropic shell elements, (2) the upper and lower skin are designed with the same element, and (3) the leading-spar and the trailing-edge spar are designed with the same element. Thus, the updating is to figure out the optimal combination of width w_{box}, height h_{box}, skin thickness t_1, spar thickness t_2, tip rib thickness t_3, material properties E_{box}, μ_{box}, and ρ_{box} of the wing box, as shown in Figure 8. Eight design variables are involved in this problem. By

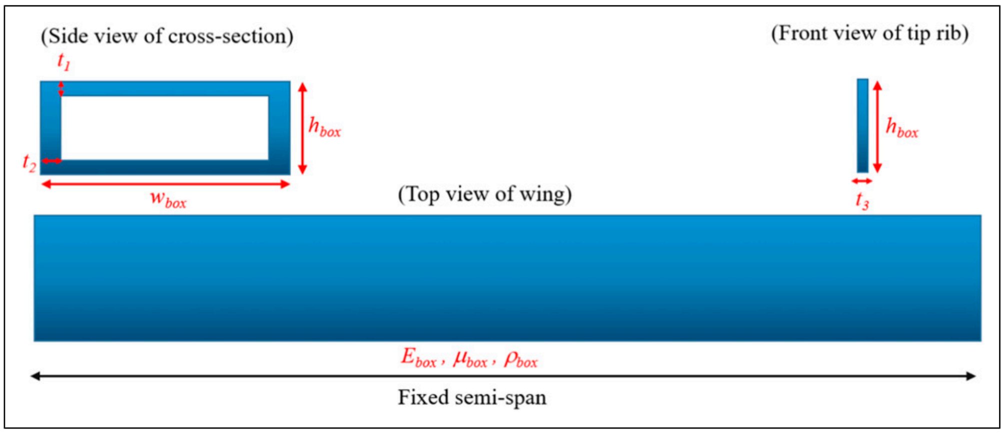

Figure 8. Design variables of the wing box.

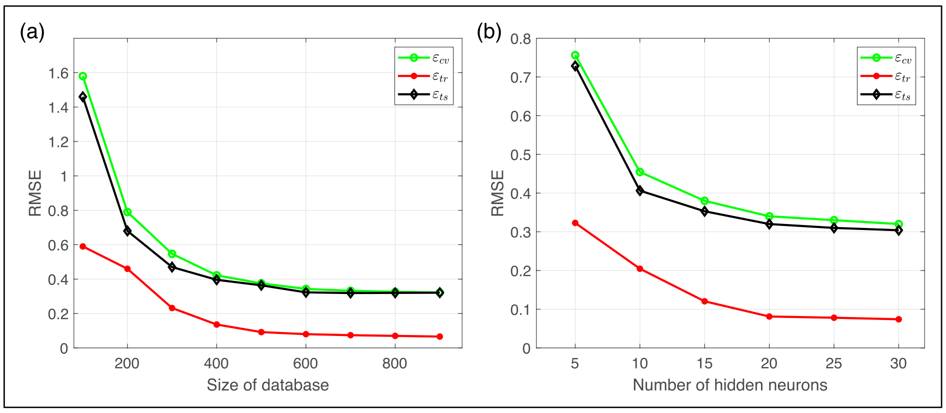

Figure 9. Performance of box's neural networks with different settings.

setting the tolerances as 0.1 and 0.05, equation (3) is formulated as

$$\min_{\mathbf{x}_{box}} \sum \varepsilon(\boldsymbol{x}_{box})$$

$$\boldsymbol{x}_{box} = \{w_{box}, h_{box}, t_1, t_2, t_3, E_{box}, \mu_{box}, \rho_{box}\}$$

$$\begin{cases} \varepsilon_{f_i} \leq 0.1 & \\ \varepsilon_{\varphi_i} \leq 0.1 & i = 1,2,3,4 \\ \varepsilon_m \leq 0.05 & \\ \boldsymbol{x}_{box}^{lb} \leq \boldsymbol{x}_{box} \leq \boldsymbol{x}_{box}^{ub} & \end{cases} \tag{10}$$

Generated wing box model. Depending on the discretization, the wing box model consists of hundreds of CQUAD4 elements. Nine neural networks with varying sample sizes and six other neural networks with different numbers of hidden neurons are conducted for trial studies. Figure 9 illustrates the neural network performance in terms of the training error and RMSE of testing and 10-fold cross-validation. Compared to the updating problem from beam to thin shell model, three more input components are involved in this problem, making the converged RMSE larger. Six hundred samples and 20 hidden neurons in the first hidden layer are then required to achieve a good approximation.

Therefore, an initial neural network with 600 samples (480 for training and 120 for testing) and a refined neural network with 300 samples (240 for training and 60 for testing) are established to generate the wing box model. Both neural networks are composed of 20 neurons in the first hidden layer. Within the design spaces shown in Table 9, training and testing results of these box's neural networks are plotted in Figure 10.

By comparing the relevant wing boxes and the original beam, individual errors in natural frequencies and masses are shown in Tables 10 and 11. Model shape errors are shown in Table 12. Mass and natural frequency errors of

Table 9. Design space of wing box's neural networks.

	Initial box's neural network	Refined box's neural network
w_{box}, m	0.6–1.2	0.75–0.85
h_{box}, m	0.15–0.3	0.2–0.3
t_1,m	0.005–0.015	0.07–0.012
t_2,m	0.003–0.01	0.003–0.008
t_3,m	0.005–0.015	0.005–0.015
E_{shell}, GPa	150–400	150–400
μ_{box}	0.25–0.45	0.25–0.45
ρ_{box},kg/m^3	1100–2700	1100–2700

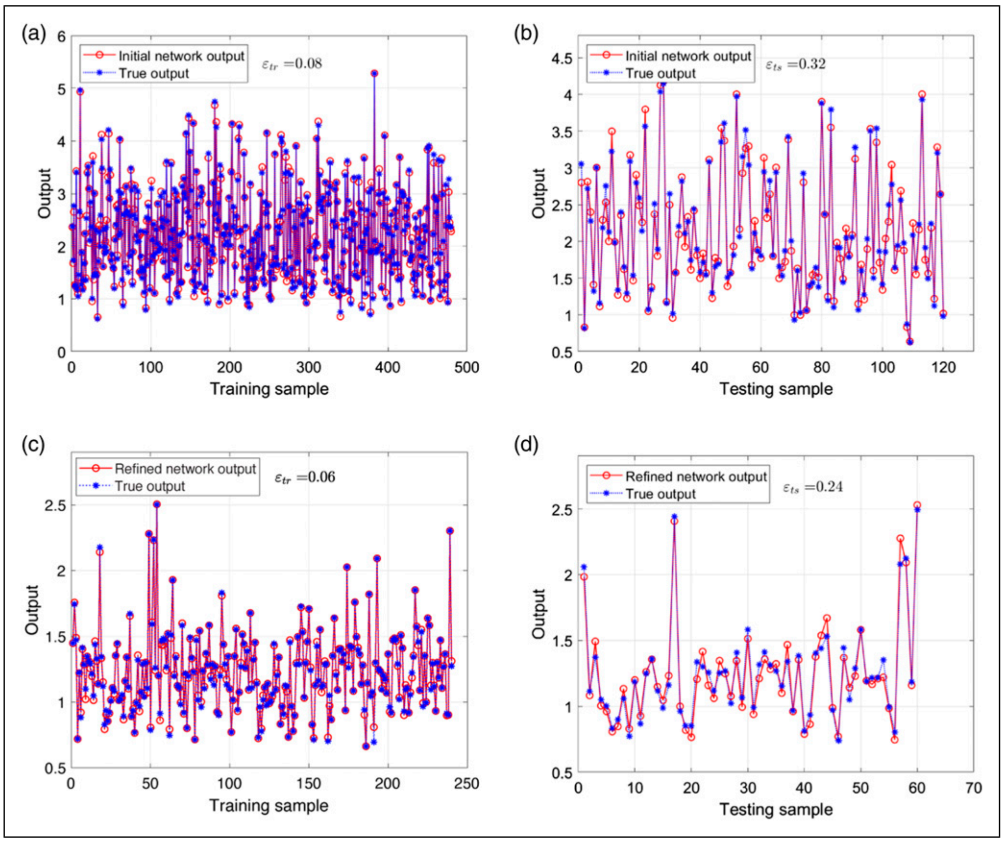

Figure 10. Comparison of box's neural networks.

Table 10. Frequencies and masses of initially generated wing box model.

	Value	Relative error
1st out-of-plane bending, Hz	8.27	6.62×10^{-2}
2nd out-of-plane bending, Hz	47.83	3.08×10^{-2}
1st in-plane bending, Hz	17.19	3.58×10^{-3}
1st torsion, Hz	82.59	4.36×10^{-2}
Mass, kg	230.30	4.04×10^{-2}

Table 11. Frequencies and masses of refined wing box model.

	Value	Relative error
1st out-of-plane bending, Hz	8.02	3.30×10^{-2}
2nd out-of-plane bending, Hz	46.56	5.65×10^{-2}
1st in-plane bending, Hz	17.55	1.02×10^{-3}
1st torsion, Hz	81.91	3.50×10^{-2}
Mass, kg	232.69	3.33×10^{-2}

the wing box related to the initial neural network satisfy their inequality constraints. However, the resulting torsional mode shape error exceeds its tolerance. In the refinement, the accuracy of the second out-of-plane bending and first in-plane bending modes' frequencies are sacrificed for the accuracy of the torsional mode.

Table 12. Mode shape errors of generated wing box models.

	Initial box	Refined box
1st out-of-plane bending	7.26×10^{-3}	8.26×10^{-3}
2nd out-of-plane bending	4.89×10^{-2}	7.13×10^{-2}
1st in-plane bending	1.52×10^{-3}	1.56×10^{-3}
1st torsion	1.10×10^{-1}	7.11×10^{-2}

Modifications of mode shape errors can also be seen from eigenvectors, shown in Figures 11–14. For the bending modes, both the translational and rotational components are compared. For the torsional mode, only the rotational component is compared since the translational eigenvectors are negligible. Discrepancies can be observed in the second out-of-plane bending mode and the first torsion mode, in which the mode shape errors are in the magnitude of 10^{-2}. In this problem, such discrepancies can be obtained by the RMSE but not the modal assurance criteria (MAC). The MAC may provide a value of 0.99 for the comparison between the original beam and the initial wing box model. The 0.99 indicates a good match which is not the case. Note that the tip rib may change the boundary constraint of the wing box such that the maximum rotations do not occur exactly at the tip. The one-cell wing box model that matches the original beam is constructed with the geometry of w_{box} = 0.75 m, t_{box} = 0.24 m, t_1 = 0.0085 m, t_2 = 0.0045 m, t_3 = 0.0086 m, and the material

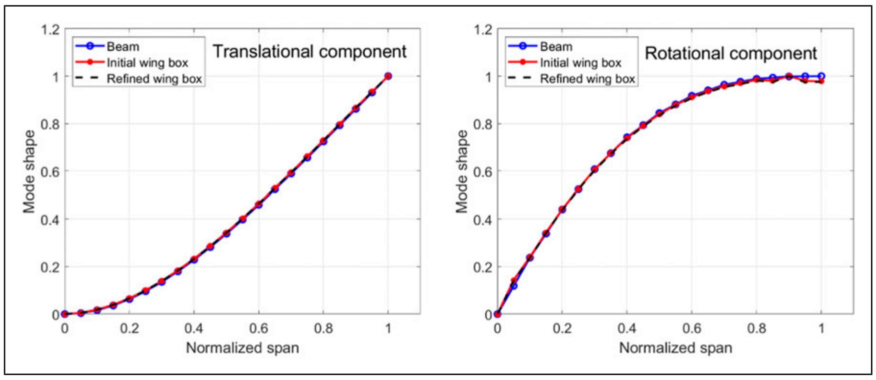

Figure 11. Comparison of the first out-of-plane bending mode shape.

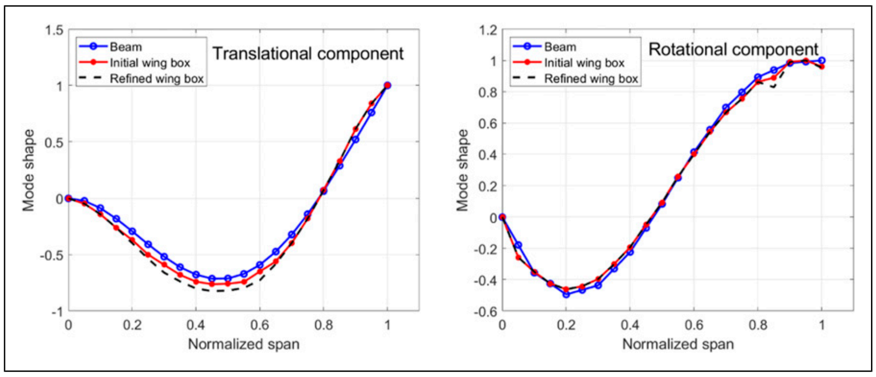

Figure 12. Comparison of the second out-of-plane bending mode shape.

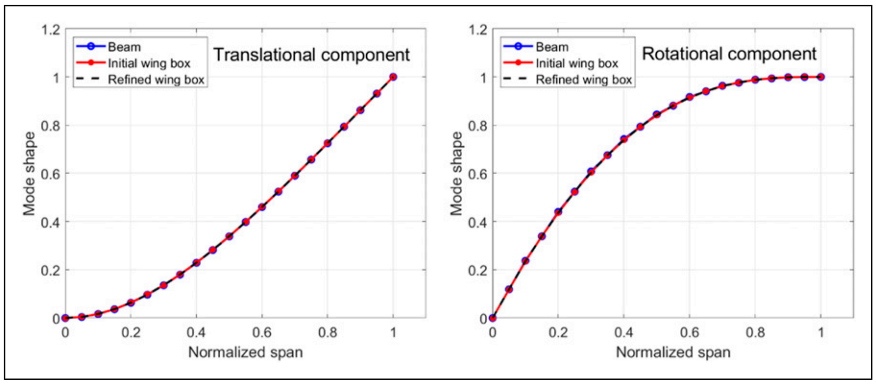

Figure 13. Comparison of the first in-plane bending mode shape.

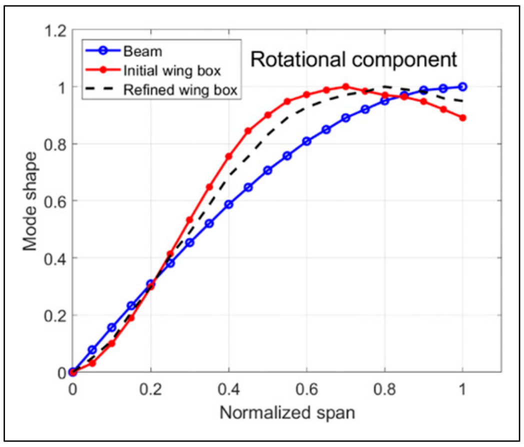

Figure 14. Comparison of the torsional mode shape.

properties are E_{box} = 270 GPa, μ_{box} = 0.31, and ρ_{box} =1570 kg/m^3.

Conclusions

A novel approach for model updating using neural networks has been successfully demonstrated using simulated data. This approach enables structural updating from the lower fidelity to the higher fidelity. A substantial advantage of this technique is its capability for dimensional expansion with no derivatives. The main drawback is that the generality is restricted by the architecture of the feed-forward neural network. However, it can be seen that there is significant potential for this updating approach with a more comprehensive database.

In this approach, the desired higher-fidelity design was viewed as an unknown function of dynamic properties of the structure obtained by the finite element solver. In doing so, a tool was developed to automate sampling processes, generating MSC.Nastran simulation bulk file, initiating the modal solver, collecting data, and computing error vectors. The feasibility of the feed-forward neural network was demonstrated by expanding beam models to shell and wing box models. The selection of neural network design is also covered through the numerical cases.

The training process demanded computationally expensive finite element solutions for building the database, especially for collecting data between MATLAB and MSC.Nastran. Efforts can be made to develop a parallel process enclosed in MATLAB. With further improvement, the neural network approach also shows significant potential to involve more practical wing box models with internal configurations.

As only SOL 103 was performed with the MSC.Nastran (high-fidelity) models, only the linear modal characteristics were considered in the current study. Even though additional nonlinear characteristics can be further considered, it is out of the scope of this paper. It is also important to note that the updating process from a low-fidelity model to a high-fidelity model is a dimension expansion problem, which may result in infinite solutions. In order to avoid such a problem, this study applied topological information, for example, the high-fidelity model is either plate-like or made of a box. As such topological information is usually available in practice, this treatment is reasonable and feasible.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The research was financially supported by NASA Aeronautics Research Mission Directorate (ARMD) under contract NNX15AV41A, with the technical monitor Jeffrey K. Viken.

ORCID iD

Weihua Su https://orcid.org/0000-0002-4458-0524

References

1. Ciarlet P and Destuynder P. A justification of a nonlinear model in plate theory. *Computer Methods Appl Mech Eng* 1979; 17-18(1): 227–258, doi:10.1016/0045-7825(79)90089-6.
2. Berdichevskii V. Variational-asymptotic method of constructing a theory of shells. *J Appl Math Mech* 1979; 43(4): 711–736. DOI: 10.1016/0021-8928(79)90157-6.
3. Hodges D, Atilgan A, Cesnik C, et al. On a simplified strain energy function for geometrically nonlinear behaviour of anisotropic beams. *Composites Eng* 1992; 2(5–7): 513–526. DOI: 10.1016/0961-9526(92)90040-D.
4. Cesnik C and Hodges D. VABS: A new concept for composite rotor blade cross-sectional modeling. *J Am Helicopter Soc* 1997; 42(1): 27–38, doi:10.4050/JAHS.42.27.
5. Yu W, Hodges D, Volovoi V, et al. On timoshenko-like modeling of initially curved and twisted composite beams. *Int J Sol Structures* 2002; 39(19): 5101–5121. DOI: 10.1016/S0020-7683(02)00399-2.
6. Yu W, Hodges D and Ho J. Variational asymptotic beam sectional analysis – An updated version. *Int J Eng Sci* 2012; 59: 40–64, doi:10.1016/j.ijengsci.2012.03.006.
7. Levin R and Lieven N. Dynamic finite element model updating using neural networks. *J Sound Vibration* 1998; 210(5): 593–607. DOI: 10.1006/jsvi.1997.1364.
8. Jaishi B and Ren W. Finite element model updating based on eigenvalue and strain energy residuals using multiobjective optimisation technique. *Mech Syst Signal Process* 2007; 21(5): 2295–2317, doi:10.1016/j.ymssp.2006.09.008.
9. Lu Y and Tu Z. A two-level neural network approach for dynamic fe model updating including damping. *J Sound Vibration* 2004; 275(3–5): 931–952. DOI: 10.1016/S0022-460X(03)00796-X.
10. Yin T, Zhu H and Fu S. Damage identification of periodically-supported structures following the bayesian probabilistic approach. *Int J Struct Stab Dyn* 2019; 19(01): 1940011, doi:10.1142/S021945541940011X.
11. Mottershead J, Link M and Friswell M. The sensitivity method in finite element model updating: A tutorial. *Mech Syst Signal Process* 2011; 25(7): 2275–2296, doi:10.1016/j.ymssp.2010.10.012.
12. Abdeljaber O, Avci O, Kiranyaz S, et al. Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks. *J Sound Vibration* 2017; 388: 154–170, doi:10.1016/j.jsv.2016.10.043.
13. Yin T and Zhu H. An efficient algorithm for architecture design of bayesian neural network in structural model updating. *Computer-Aided Civil Infrastructure Eng* 2020; 35(4): 354–372. DOI: 10.1111/mice.12492.
14. Ismail H, Singh M, Darwish S, et al. Developing ANN-Kriging hybrid model based on process parameters for prediction of mean residence time distribution in twin-screw wet granulation. *Powder Tech* 2019; 343: 568–577, doi:10.1016/j.powtec.2018.11.060.
15. Cybenko G. Approximation by superpositions of a sigmoidal function. *Math Control Signals Syst* 1989; 2: 303–314, doi:10.1007/BF02551274.
16. Liu Z, Fang H and Xu J. Identification of piecewise linear dynamical systems using physically-interpretable neural-fuzzy networks: methods and applications to origami structures. *Neural Networks* 2019; 116: 74–87. DOI: 10.1016/j.neunet.2019.04.007.
17. Yu W and Hodges D. Elasticity solutions versus asymptotic sectional analysis of homogeneous, isotropic, prismatic beams. *J Appl Mech* 2004; 71(1): 15–23, doi:10.1115/1.1640367.
18. Hagan M, Demuth H, Beale M, et al. *Neural network design*. 2nd ed.. Martin Hagan, 2014.
19. Sola J and Sevilla J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Trans Nucl Sci* 1997; 44(3): 1464–1468, doi:10.1109/23.589532.
20. Bishop C. *Neural networks for pattern recognition*. Oxford University Press, USA, 1996.
21. Magar K, Reich G, Rickey M, et al. Aerodynamic characteristics prediction via artificial hair sensor and feedforward neural network. *ASME Conference on Smart Mater Adaptive Structures Intell Syst*. Colorado Springs, CO: American Society of Mechanical Engineers, p. V002T06A004, doi:10.1115/SMASIS2015-8890.
22. Wang Q, Medeiros R, Cesnik C, et al. Techniques for improving neural network-based aerodynamics reduced-order models. *AIAA SciTech 2019 Forum*. San Diego, CA: American Institute of Aeronautics and Astronautics, doi:10.2514/6.2019-1849.