Formulation and Application of SMU – an Open-Source MATLAB Package for Structural Model Updating

Yu Otsuki[1], Peter Lander[1], Xinjun Dong[1], Yang Wang[1, 2]

[1] *School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA, USA*

[2] *School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA*

**yang.wang@ce.gatech.edu*

Abstract: This paper describes a MATLAB package for structural model updating, named SMU. The SMU package updates parameter values of a finite element model by solving optimization problems utilizing modal properties obtained from sensor measurements. In particular, the package offers three model updating formulations, namely (1) the MAC (modal assurance criterion) value approach, (2) the eigenvector difference approach, and (3) the modal dynamic residual formulation. The first two belong to the family of modal property difference formulations. For each formulation, the analytical Jacobian derivative of the objective function is derived and implemented in SMU. Since the formulated optimization problems are generally nonconvex, the global optimality of the solution cannot be guaranteed using off-the-shelf optimization algorithms. In order to increase the chance of finding a better local minimum, the SMU package can perform gradient search from randomly generated starting points. Several examples for the model updating of as-built structures are included in the GitHub package. This paper demonstrates the SMU functionality through model updating of an 18-DOF model and a concrete building frame model.

Keywords: finite element model updating, open-source software, non-convex optimization, optimization algorithms, modal analysis

1 Introduction

In modern structural analysis, a great amount of effort has been devoted to finite element (FE) modeling towards simulating the behavior of an as-built structure. In general, predictions by FE models often differ from in-situ experimental measurements. Various inaccuracies in the models can contribute to the discrepancy. For example, idealized connections and support conditions are commonly used in structural analysis and design, while these

conditions with infinite or zero stiffness do not exist in reality. In addition, material properties of an as-built structure are usually different from the nominal values, particularly for concrete. To obtain a more accurate FE model that truly reflects behaviors of an as-built structure, data collected from the in-situ experiments can be used to update the values of selected model parameters (e.g. support stiffness or mass parameters) in the FE model. This process is known as FE model updating. An updated FE model can more accurately predict structural behavior under various loading conditions and also serve as the baseline for identifying structural property changes over time, potentially due to deterioration. In the meantime, benefiting from the development of low-cost wireless sensing systems (Lynch and Loh, 2006; Kane et al., 2014; Dong et al., 2016), more and more structural sensors are available for measuring structural responses. As a result, large amounts of sensor data collected from as-built structures are becoming available for FE model updating, which on the other hand, poses computational challenges for performing model updating.

Numerous FE model updating algorithms have been developed and practically applied in the past few decades (Friswell and Mottershead, 1995). Most algorithms can be categorized into two groups, i.e. time-domain and frequency-domain approaches. Time-domain approaches deal with time history data collected from the actual structure directly, without the requirement for extracting modal properties (Yang et al., 2006; Doucet and Tadić, 2003; Chatzi and Smyth, 2009; Sato and Qi, 1998). Overall, the time-domain approaches suffer convergence difficulties and high computational cost when applied to large-scale FE models. Different from the time-domain approaches, the frequency-domain approaches can update an FE model using frequency-domain modal properties extracted from experimental measurements. These include resonant frequencies, vibration mode shapes and damping ratios. In particular, early researchers started by minimizing an objective function consisting of the differences between measured and simulated resonant frequencies. Naturally, this is formulated as a mathematical optimization problem with the selected structural parameters as optimization variables. This category of model updating approaches is named the modal property difference formulation. For example, Zhang et al. (2000) proposed an eigenvalue sensitivity-based model updating approach that was applied on a scaled suspension bridge model, and the updated FE model shows resonant frequencies closer to the experimental measurements. Salawu (1997) reviewed various model updating algorithms using resonant frequency differences, and concluded that differences in frequencies may not be sufficient enough for accurately identifying structural parameter values. As a result, other modal properties, e.g. mode shapes or modal flexibility, were investigated for model updating. For example, Moller and Friberg (1998) adopted the modal assurance criterion (MAC)-related function for updating the model of an industrial structure, in attempt to make the updated FE model

generate mode shapes that are closer to those extracted from experimental measurements. FE model updating using differences in simulated and experimental mode shapes and frequencies was also applied to damage assessment of a reinforced concrete beam (Teughels et al., 2002). Yuen (2012) developed an efficient model updating algorithm using frequencies and mode shapes at only a few selected degrees-of-freedom (DOFs) for the first few modes. Another formulation widely applied in frequency domain model updating is the modal dynamic residual formulation, which minimizes the residuals of generalized eigenvalue problems involving stiffness and mass matrices (Farhat and Hemez, 1993; Li et al., 2018; Kosmatka and Ricles, 1999). The model updating formulations are in general nonconvex, and global optimality of the solution cannot be guaranteed using off-the-shelf local optimization algorithms (Wang et al., 2019a). Attempts have been made to obtain the global optimum of the optimization problems in FE model updating (Li et al., 2018; Otsuki et al., 2021a; Bakir et al., 2008; Hofmeister et al., 2019). Furthermore, model updating of large-scale structures is computationally challenging. Substructure-based approaches have been proposed to reduce the computational cost (Weng et al., 2011; Zhu et al., 2016; Zhu et al., 2021).

From a practical point of view, the implementation of structural model updating can be challenging especially for those who are not familiar with mathematical optimization. Difficulties arise in how to construct an objective function and variable constraints, how to select optimization algorithms, and how to deal with nonconvex problems. Optimization algorithms perform gradient search using the Jacobian derivative at every iteration. It has been known that the analytical gradient can achieve better accuracy in model updating compared to the numerical gradient (Dong and Wang, 2018). However, the derivation and implementation of analytical gradients for various model updating formulations can be challenging.

SMU, an open-source MATLAB package for structural model updating, is developed and shared with the research community (Wang et al., 2019b). The package implements three model updating formulations in the frequency domain with the corresponding analytical gradients. Currently, SMU supports optimization algorithms available in the MATLAB optimization toolbox (MathWorks Inc., 2019), such as the Levenberg-Marquardt algorithm, the trust-region-reflective algorithm, the interior-point method, *etc*. In order to increase the chance of finding a better local minimum for the nonconvex optimization problem, the package performs gradient search from randomly generated starting points. Several examples for the model updating of as-built structures are included in the GitHub package. SMU has been readily utilized to facilitate research in structural model updating (Zhang and Sun, 2020; Otsuki et al., 2021a; Dong and Wang, 2019).

The rest of the paper is organized as follows. Section 2 presents the three model updating formulations adopted in SMU. Section 3 shows the mathematical derivation of the analytical Jacobian derivatives for each model updating formulation. Section 4 describes the key components and user interface of SMU. Sections 5 and 6 demonstrate the model updating of an 18-story steel frame and a concrete building frame using SMU. Finally, Section 7 provides a summary and conclusions.

2 Structural model updating formulations

The formulations adopted in the current version of SMU update stiffnesses of structural models, while if needed, the functionality can be extended for mass and damping updating. For model updating of a linear structure with *N* DOFs, the stiffness matrices can be expressed as:

$$\mathbf{K}(\boldsymbol{\alpha}) = \mathbf{K}_0 + \sum_{j=1}^{n_{\boldsymbol{\alpha}}} \alpha_j \mathbf{K}_j \tag{1}$$

where $\mathbf{K}_0 \in \mathbb{R}^{N\times N}$ is the initial stiffness matrix; $n_{\boldsymbol{\alpha}}$ is the number of stiffness updating variables; α_j is the *j*-th entry of the stiffness updating vector variable $\boldsymbol{\alpha} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$, which represents the relative change of a stiffness parameter from the initial value; and $\mathbf{K}_j \in \mathbb{R}^{N\times N}$ is the influence matrix for the *j*-th stiffness updating variable α_j.

The formulations adopted in SMU are based on the generalized eigenvalue problem in structural dynamics:

$$[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i \mathbf{M}]\{\boldsymbol{\psi}_i\} = \mathbf{0}, \qquad i = 1 \dots n_{\text{modes}} \tag{2}$$

where $\mathbf{M} \in \mathbb{R}^{N\times N}$ denotes the mass matrix; $\lambda_i \in \mathbb{R}$ and $\boldsymbol{\psi}_i \in \mathbb{R}^N$ are the *i*-th eigenvalue and eigenvector, respectively; and n_{modes} denotes the number of modes. Note that λ_i and $\boldsymbol{\psi}_i$ implicitly depend on $\boldsymbol{\alpha}$, and thus can be denoted as $\lambda_i(\boldsymbol{\alpha})$ and $\boldsymbol{\psi}_i(\boldsymbol{\alpha})$. In general, eigenvectors obtained through field testing are limited to the DOFs measured by sensors. To reflect the measured DOFs, we assume the DOFs are rearranged properly and define $\boldsymbol{\psi}_i(\boldsymbol{\alpha}) = [\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha}); \ \boldsymbol{\psi}_i^{\mathcal{U}}(\boldsymbol{\alpha})] \in \mathbb{R}^N$, where $\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha}) \in \mathbb{R}^{n_{\mathcal{M}}}$ represents the eigenvector entries corresponding to the measured DOFs and $\boldsymbol{\psi}_i^{\mathcal{U}} \in \mathbb{R}^{n_{\mathcal{U}}}$ corresponds to the unmeasured DOFs. In this paper, we use a semicolon ";" to concatenate vectors/matrices by columns. Note that $n_{\mathcal{M}} + n_{\mathcal{U}} = N$, the total number of DOFs. In this paper, experimentally obtained eigenvalues and eigenvectors are denoted as λ_i^{EXP} and $\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}$ (at measured DOFs), respectively.

2.1 Modal property difference formulation #1: MAC value approach

The SMU package supports the MAC (modal assurance criterion) value approach, which is categorized as a modal property difference formulation.

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \sum_{i=1}^{n_{\text{modes}}} \left\{ \left(\frac{\lambda_i^{\text{EXP}} - \lambda_i(\boldsymbol{\alpha})}{\lambda_i^{\text{EXP}}} \cdot w_{\lambda_i} \right)^2 + \left(\frac{1 - \sqrt{\text{MAC}_i(\boldsymbol{\alpha})}}{\sqrt{\text{MAC}_i(\boldsymbol{\alpha})}} \cdot w_{\boldsymbol{\psi}_i} \right)^2 \right\} \tag{3}$$

$$\text{subject to} \quad \mathbf{L}_{\boldsymbol{\alpha}} \leq \boldsymbol{\alpha} \leq \mathbf{U}_{\boldsymbol{\alpha}}$$

where w_{λ_i} represents the weighting factor of the *i*-th eigenvalue difference; $w_{\boldsymbol{\psi}_i}$ represents the weighting factor of the *i*-th MAC value difference; and $\mathbf{L}_{\boldsymbol{\alpha}}$ and $\mathbf{U}_{\boldsymbol{\alpha}} \in \mathbb{R}^{n_{\boldsymbol{\alpha}}}$ denote the lower and upper bounds for the variable $\boldsymbol{\alpha}$, respectively. $\text{MAC}_i(\boldsymbol{\alpha})$ represents the modal assurance criterion between the *i*-th experimental and simulated eigenvectors/mode shapes at measured DOFs, i.e. $\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}$ and $\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})$.

$$\text{MAC}_i(\boldsymbol{\alpha}) = \frac{\left(\left(\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \right)^{\text{T}} \boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha}) \right)^2}{\left\| \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \right\|_2^2 \left\| \boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha}) \right\|_2^2}, \qquad i = 1 \ldots n_{\text{modes}} \tag{4}$$

Here $\|\cdot\|_2$ denotes the $\mathcal{L}_2$-norm of a vector. The MAC value represents the similarity between two vectors. When the two vectors are collinear, the MAC value is 1. When two vectors are orthogonal, the MAC value is 0.

At every iteration of an optimization gradient search, the generalized eigenvalue problem in Eq. (2) is solved using values of $\boldsymbol{\alpha}$ at the current iteration, which produces simulated eigenvalues $\lambda_i(\boldsymbol{\alpha})$ and eigenvectors $\boldsymbol{\psi}_i(\boldsymbol{\alpha})$. This process determines that with implicit functions $\lambda_i(\boldsymbol{\alpha})$ and $\text{MAC}_i(\boldsymbol{\alpha})$, the objective function in Eq. (3) has to remain implicit. For $\boldsymbol{\alpha}$ at the current iteration, the optimization algorithm then evaluates the objective function value in Eq. (3) and calculates the Jacobian derivative to find the next search gradient. It then moves to the next iteration and repeats this process until convergence criteria are satisfied. Besides having an implicit objective function, the MAC value approach provides a nonconvex optimization problem for which the global optimality of the solution cannot be guaranteed using local optimization algorithms.

2.2 Modal property difference formulation #2: eigenvector difference approach

Another modal property difference formulation supported in SMU is the eigenvector difference approach.

$$\underset{\boldsymbol{\alpha}}{\text{minimize}} \sum_{i=1}^{n_{\text{modes}}} \left\{ \left(\frac{\lambda_i^{\text{EXP}} - \lambda_i(\boldsymbol{\alpha})}{\lambda_i^{\text{EXP}}} \cdot w_{\lambda_i} \right)^2 + \left\| \left\{ \boldsymbol{\psi}_{-q_i,i}^{\text{EXP},\mathcal{M}} - \boldsymbol{\psi}_{-q_i,i}^{\mathcal{M}}(\boldsymbol{\alpha}) \right\} \cdot w_{\boldsymbol{\psi}_i} \right\|_2^2 \right\} \quad (5)$$

$$\text{subject to} \quad \mathbf{L}_{\boldsymbol{\alpha}} \leq \boldsymbol{\alpha} \leq \mathbf{U}_{\boldsymbol{\alpha}}$$

where w_{λ_i} represents the weighting factor of the i-th eigenvalue difference; $w_{\boldsymbol{\psi}_i}$ represents the weighting factor of the i-th eigenvector difference. The experimental eigenvector at measured DOFs $\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}$ is normalized so that the largest entry of $\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}$, denoted as the q_i-th entry, equals to 1. Accordingly, the simulated eigenvector at measured DOFs $\boldsymbol{\psi}_i^{\mathcal{M}}$ is also normalized so that the q_i-th entry equals to 1. $\boldsymbol{\psi}_{-q_i,i}^{\text{EXP},\mathcal{M}}$ and $\boldsymbol{\psi}_{-q_i,i}^{\mathcal{M}} \in \mathbb{R}^{n_{\mathcal{M}}-1}$ represent the eigenvectors at the measured DOFs with the q_i-th entry removed. Compared to the MAC value approach, the eigenvector difference approach directly minimizes the differences between the entries in the experimental and simulated eigenvectors. Note that the eigenvector difference approach also has an implicit objective function and is a nonconvex optimization problem.

2.3 Modal dynamic residual formulation

The last formulation currently implemented in SMU is the modal dynamic residual formulation. The formulation minimizes the residual of generalized eigenvalue equations in structural dynamics. The residuals are calculated based on the stiffness and mass matrices in combination with experimentally obtained eigenvalues and eigenvectors.

$$\begin{aligned} &\underset{\boldsymbol{\alpha},\boldsymbol{\psi}^{\mathcal{U}}}{\text{minimize}} \sum_{i=1}^{n_{\text{modes}}} \left\| \left[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i^{\text{EXP}} \mathbf{M} \right] \begin{Bmatrix} \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_i^{\mathcal{U}} \end{Bmatrix} \cdot w_i \right\|_2^2 \\ &\text{subject to} \quad \mathbf{L}_{\boldsymbol{\alpha}} \leq \boldsymbol{\alpha} \leq \mathbf{U}_{\boldsymbol{\alpha}} \\ &\qquad \mathbf{L}_{\boldsymbol{\psi}_i^{\mathcal{U}}} \leq \boldsymbol{\psi}_i^{\mathcal{U}} \leq \mathbf{U}_{\boldsymbol{\psi}_i^{\mathcal{U}}}, \; i = 1 \ldots n_{\text{modes}} \end{aligned} \quad (6)$$

where w_i represents the weighting factor for the i-th mode, and $\boldsymbol{\psi}_i^{\mathcal{U}} \in \mathbb{R}^{n_{\mathcal{U}}}$ represents the unmeasured entries of i-th eigenvector. Compared to the modal property difference formulations, the modal dynamic residual formulation has more updating variables $\boldsymbol{\psi}_i^{\mathcal{U}}, i = 1 \ldots n_{\text{modes}}$. Note that the objective function in Eq. (6) is explicitly expressed by optimization variables $\boldsymbol{\alpha}$ and $\boldsymbol{\psi}_i^{\mathcal{U}}$, while Eq. (3) and Eq. (5) for the modal property difference formulations are not. The modal dynamic residual formulation is convex in the rare case when all DOFs are measured (i.e. $n_{\mathcal{M}} = N$, $n_{\mathcal{U}} = 0$, and $\boldsymbol{\psi}_i^{\mathcal{U}}$ is null). However, when only some DOFs are measured, as often the case in practice, the formulation is nonconvex.

2.4 Summary of model updating formulations

Each formulation implemented in SMU has pros and cons in terms of computational efficiency and updating performance. In general, by not involving the MAC value term, the eigenvector difference approach (Eq. (5)) is computationally more efficient than the MAC value approach (Eq. (3)). Numerical studies have shown that the eigenvector difference approach usually achieves higher accuracy than the MAC value approach (Dong and Wang, 2018). The modal dynamic residual formulation (Eq. (6)) is based on a different concept from the MAC value and the eigenvector difference approaches. The modal dynamic residual formulation does not directly minimize the discrepancy of modal properties, and thus, the updated model may have modal properties that are more different from experimental values. It is highly recommended to investigate and compare model updating results using different formulations especially when dealing with experimental data.

In general, when using experimental data, the system identification results of eigenvectors have more variability and can be less accurate compared to eigenvalues. Moreover, it is more challenging to accurately identify the eigenvalues and eigenvectors for higher modes than those for lower modes. Therefore, users can take into account the reliability of the identified modal parameters through the use of weighting factors, w_{λ_i} and $w_{\boldsymbol{\psi}_i}$ in Eq. (3) and Eq. (5) or w_i in Eq. (6). Furthermore, if several vibration records and system identification results are available, statistical properties of identified results, such as standard deviation, can be reflected in weighting factors to consider uncertainties in measurements and system identification results.

3 Jacobian derivatives of the formulations

The numerical optimization algorithms solving Eq. (3), Eq. (5), and Eq. (6) are iterative. At every iteration step, the Jacobian derivative (short-named as the Jacobian) of the objective function is often used to determine the search direction. To facilitate the Jacobian derivation, the optimization problems in Eq. (3), Eq. (5), and Eq. (6) can be rewritten as:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} \quad f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^{\mathrm{T}}\mathbf{r}(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|_2^2 \\ &\text{subject to} \quad \mathbf{L}_{\mathbf{x}} \le \mathbf{x} \le \mathbf{U}_{\mathbf{x}} \end{aligned} \tag{7}$$

where $\mathbf{x} \in \mathbb{R}^{n_{\mathbf{x}}}$ is a vector variable; $\mathbf{r}(\mathbf{x}): \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}^{n_{\mathbf{r}}}$ is a residual vector function; $f(\mathbf{x}): \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}$ is an objective function; $n_{\mathbf{x}}$ is the length of the vector variable $\mathbf{x}$; and $n_{\mathbf{r}}$ is the length of the residual vector function $\mathbf{r}$. The Jacobian

of the objective function $f(\mathbf{x})$ is defined as $\mathrm{D}_{\mathbf{x}}f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_{n_{\mathbf{x}}}} \end{bmatrix} \in \mathbb{R}^{1\times n_{\mathbf{x}}}$. In the following subsections, the Jacobians of the objective functions shown in Eq. (3), Eq. (5), and Eq. (6) will be derived.

3.1 Jacobian of modal property difference formulation #1: MAC value approach

When using the MAC value approach in Eq. (3), the vector variable **x** in Eq. (7) is the stiffness updating vector variable $\boldsymbol{\alpha}$. Hence, the MAC value formulation in Eq. (3), is rewritten in least squares form as $f(\boldsymbol{\alpha}) = \mathbf{r}(\boldsymbol{\alpha})^{\mathrm{T}}\mathbf{r}(\boldsymbol{\alpha})$ with a residual vector function $\mathbf{r}(\boldsymbol{\alpha})$: $\mathbb{R}^{n_{\boldsymbol{\alpha}}} \to \mathbb{R}^{2\cdot n_{\text{modes}}}$ as:

$$\mathbf{r}(\boldsymbol{\alpha}) = \begin{bmatrix} \mathbf{r}_1(\boldsymbol{\alpha}) \\ \vdots \\ \mathbf{r}_{n_{\text{modes}}}(\boldsymbol{\alpha}) \end{bmatrix} = \begin{bmatrix} \left(\lambda_1^{\text{EXP}} - \lambda_1(\boldsymbol{\alpha})\right)/\lambda_1^{\text{EXP}} \cdot w_{\lambda_1} \\ \left(1 - \sqrt{\text{MAC}_1(\boldsymbol{\alpha})}\right)/\sqrt{\text{MAC}_1(\boldsymbol{\alpha})} \cdot w_{\boldsymbol{\psi}_1} \\ \vdots \\ \left(\lambda_{n_{\text{modes}}}^{\text{EXP}} - \lambda_{n_{\text{modes}}}(\boldsymbol{\alpha})\right)/\lambda_{n_{\text{modes}}}^{\text{EXP}} \cdot w_{\lambda_{n_{\text{modes}}}} \\ \left(1 - \sqrt{\text{MAC}_{n_{\text{modes}}}(\boldsymbol{\alpha})}\right)\Big/\sqrt{\text{MAC}_{n_{\text{modes}}}(\boldsymbol{\alpha})} \cdot w_{\boldsymbol{\psi}_{n_{\text{modes}}}} \end{bmatrix} \tag{8}$$

where $\mathbf{r}_i(\boldsymbol{\alpha})$: $\mathbb{R}^{n_{\boldsymbol{\alpha}}} \to \mathbb{R}^2$ equals $\left[\frac{\lambda_i^{\text{EXP}} - \lambda_i(\boldsymbol{\alpha})}{\lambda_i^{\text{EXP}}} \cdot w_{\lambda_i};\quad \frac{1-\sqrt{\text{MAC}_i(\boldsymbol{\alpha})}}{\sqrt{\text{MAC}_i(\boldsymbol{\alpha})}} \cdot w_{\boldsymbol{\psi}_i}\right], i = 1 \ldots n_{\text{modes}}$. The Jacobian for $f(\boldsymbol{\alpha})$, $\mathrm{D}_{\boldsymbol{\alpha}}f \in \mathbb{R}^{1\times n_{\boldsymbol{\alpha}}}$, equals $\mathrm{D}_{\mathbf{r}}f \cdot \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}$ by using the chain rule. The first term is $\mathrm{D}_{\mathbf{r}}f = 2\mathbf{r}^{\mathrm{T}} \in \mathbb{R}^{1\times(2\cdot n_{\text{modes}})}$. The second term is $\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r} = [\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_1;\quad \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_2;\quad \cdots\quad \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_{n_{\text{modes}}}] \in \mathbb{R}^{(2\cdot n_{\text{modes}})\times n_{\boldsymbol{\alpha}}}$. Recall the definition of the MAC value in Eq. (4), each $\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i \in \mathbb{R}^{2\times n_{\boldsymbol{\alpha}}}$ can be formed as:

$$\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i = \begin{bmatrix} -\frac{w_{\lambda_i}}{\lambda_i^{\text{EXP}}} \cdot \mathrm{D}_{\boldsymbol{\alpha}}\left(\lambda_i(\boldsymbol{\alpha})\right) \\ \left(\frac{-w_{\boldsymbol{\psi}_i}}{\sqrt{\text{MAC}_i(\boldsymbol{\alpha})}}\right)\left(\frac{\left(\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}\right)^{\mathrm{T}}}{\left(\boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}}\right)^{\mathrm{T}}\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})} - \frac{\left(\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})\right)^{\mathrm{T}}}{\left\|\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})\right\|_2^2}\right)\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})\right) \end{bmatrix}, i = 1 \ldots n_{\text{modes}} \tag{9}$$

The formulation for $\mathrm{D}_{\boldsymbol{\alpha}}\left(\lambda_i(\boldsymbol{\alpha})\right) \in \mathbb{R}^{1\times n_{\boldsymbol{\alpha}}}$ and $\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})\right) \in \mathbb{R}^{n_{\mathcal{M}}\times n_{\boldsymbol{\alpha}}}$ have been well studied by researchers (Fox and Kapoor, 1968; Nelson, 1976). Nevertheless, a simplified way of obtaining $\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})\right)$ based on the normalization of $\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})$ is presented, without expressing the derivative as a linear combination of all the eigenvectors (as in (Nelson, 1976)). Recall the generalized eigenvalue equation for the *i*-th mode:

$$[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i\mathbf{M}]\{\boldsymbol{\psi}_i\} = \mathbf{0} \tag{10}$$

By differentiating Eq. (10) with respect to the *j*-th updating variable, α_j, the following equation can be obtained.

$$[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i \mathbf{M}] \frac{\partial \boldsymbol{\psi}_i}{\partial \alpha_j} = \frac{\partial \lambda_i}{\partial \alpha_j} \mathbf{M} \boldsymbol{\psi}_i - \mathbf{K}_j \boldsymbol{\psi}_i \tag{11}$$

Assume that the eigenvalues are distinct and define the modal mass of the *i*-th mode as $m_i = (\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{M} \boldsymbol{\psi}_i$. Then, pre-multiply Eq. (11) by $(\boldsymbol{\psi}_i)^{\mathrm{T}}$, and note $(\boldsymbol{\psi}_i)^{\mathrm{T}}[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i \mathbf{M}] = \mathbf{0}$. Eq. (11) can be simplified as follows,

$$(\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{K}_j \boldsymbol{\psi}_i = \frac{\partial \lambda_i}{\partial \alpha_j} (\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{M} \boldsymbol{\psi}_i \tag{12}$$

and $\partial \lambda_i / \partial \alpha_j \in \mathbb{R}$ can be obtained.

$$\frac{\partial \lambda_i}{\partial \alpha_j} = \frac{(\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{K}_j \boldsymbol{\psi}_i}{m_i} \tag{13}$$

As a result, the Jacobian of the *i*-th simulated eigenvalue, $\mathrm{D}_{\boldsymbol{\alpha}}(\lambda_i) \in \mathbb{R}^{1 \times n_{\boldsymbol{\alpha}}}$, with respect to updating vector, $\boldsymbol{\alpha}$, can be found as follows:

$$\mathrm{D}_{\boldsymbol{\alpha}}(\lambda_i) = \begin{bmatrix} \frac{\partial \lambda_i}{\partial \alpha_1} & \frac{\partial \lambda_i}{\partial \alpha_2} & \cdots & \frac{\partial \lambda_i}{\partial \alpha_{n_{\boldsymbol{\alpha}}}} \end{bmatrix} = \begin{bmatrix} \frac{(\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{K}_1 \boldsymbol{\psi}_i}{m_i} & \frac{(\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{K}_2 \boldsymbol{\psi}_i}{m_i} & \cdots & \frac{(\boldsymbol{\psi}_i)^{\mathrm{T}} \mathbf{K}_{n_{\boldsymbol{\alpha}}} \boldsymbol{\psi}_i}{m_i} \end{bmatrix} \tag{14}$$

After obtaining $\partial \lambda_i / \partial \alpha_j$, Eq. (11) is reused to find the only remaining unknown term, $\partial \boldsymbol{\psi}_i / \partial \alpha_j \in \mathbb{R}^N$. However, $\partial \boldsymbol{\psi}_i / \partial \alpha_j$ cannot be directly obtained from Eq. (11), because $[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i \mathbf{M}]$ is rank deficient by one assuming that the eigenvalue λ_i is distinct. Nevertheless, as previously mentioned, $\boldsymbol{\psi}_i^{\mathcal{M}}$ is normalized so that the q_i-th entry always equals a constant 1. As a result, the q_i-th entry in vector $\partial \boldsymbol{\psi}_i^{\mathcal{M}} / \partial \alpha_j$ is zero, i.e. $\partial \psi_{q_i,i}^{\mathcal{M}} / \partial \alpha_j = 0$. Because of the separation by measured and unmeasured DOFs, $\boldsymbol{\psi}_i(\boldsymbol{\alpha}) = [\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha}); \quad \boldsymbol{\psi}_i^{\mathcal{U}}(\boldsymbol{\alpha})]$, the q_i-th entry in $\partial \boldsymbol{\psi}_i / \partial \alpha_j$ is also zero, i.e. $\partial \psi_{q_i,i} / \partial \alpha_j = 0$. This is utilized to resolve the rank deficiency issue of $[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i \mathbf{M}]$. Specifically, define $\mathbf{P}_i = \begin{bmatrix} \mathbf{Q}_i & \mathbf{0}_{(n_{\mathcal{M}}-1) \times n_{\mathcal{U}}} \\ \mathbf{0}_{n_{\mathcal{U}} \times n_{\mathcal{M}}} & \mathbf{I}_{n_{\mathcal{U}}} \end{bmatrix} \in \mathbb{R}^{(N-1) \times N}$ where $\mathbf{Q}_i \in \mathbb{R}^{(n_{\mathcal{M}}-1) \times n_{\mathcal{M}}}$ selects the entries of measured DOFs except for the q_i-th entry as:

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{I}_{q_i - 1} & \mathbf{0}_{(q_i-1) \times 1} & \mathbf{0}_{(q_i-1) \times (n_{\mathcal{M}} - q_i)} \\ \mathbf{0}_{(n_{\mathcal{M}} - q_i) \times (q_i - 1)} & \mathbf{0}_{(n_{\mathcal{M}} - q_i) \times 1} & \mathbf{I}_{n_{\mathcal{M}} - q_i} \end{bmatrix} \tag{15}$$

where $\mathbf{I}_{q_i-1}$ and $\mathbf{I}_{n_{\mathcal{M}}-q_i}$ denote identity matrices with size of q_i-1 and $n_{\mathcal{M}}-q_i$, respectively. Then, pre-multiplying and post-multiplying $[\mathbf{K}(\boldsymbol{\alpha})-\lambda_i\mathbf{M}]$ in Eq. (11) by $\mathbf{P}_i$ and $\mathbf{P}_i^{\mathrm{T}}$ to cross out the q_i-th row and q_i-th column, $\mathbf{B}_i \in \mathbb{R}^{(N-1)\times(N-1)}$ is generated.

$$\mathbf{B}_i = \mathbf{P}_i[\mathbf{K}(\boldsymbol{\alpha})-\lambda_i\mathbf{M}]\mathbf{P}_i^{\mathrm{T}} \tag{16}$$

Next, pre-multiply $\left(\frac{\partial\lambda_i}{\partial\alpha_j}\mathbf{M}\boldsymbol{\psi}_i - \mathbf{K}_j\boldsymbol{\psi}_i\right)$ in Eq. (11) by $\mathbf{P}_i$ to eliminate the q_i-th row and obtain $\mathbf{b}_{ij} \in \mathbb{R}^{N-1}$:

$$\mathbf{b}_{ij} = \mathbf{P}_i \cdot \left(\frac{\partial\lambda_i}{\partial\alpha_j}\mathbf{M}\boldsymbol{\psi}_i - \mathbf{K}_j\boldsymbol{\psi}_i\right) \tag{17}$$

Finally, recalling $\partial\psi_{q_i,i}/\partial\alpha_j = 0$, the elimination of the q_i-th row in Eq. (11) is equivalent to the following.

$$\mathbf{B}_i\begin{Bmatrix}\dfrac{\partial\left(\mathbf{Q}_i\boldsymbol{\psi}_i^{\mathcal{M}}\right)}{\partial\alpha_j}\\ \dfrac{\partial\left(\boldsymbol{\psi}_i^{u}\right)}{\partial\alpha_j}\end{Bmatrix} = \mathbf{b}_{ij} \tag{18}$$

Thus, the Jacobian of the *i*-th simulated eigenvector with respect to the updating variables can be shown as:

$$\begin{Bmatrix}\dfrac{\partial\left(\mathbf{Q}_i\boldsymbol{\psi}_i^{\mathcal{M}}\right)}{\partial\alpha_j}\\ \dfrac{\partial\left(\boldsymbol{\psi}_i^{u}\right)}{\partial\alpha_j}\end{Bmatrix} = \mathbf{B}_i^{-1}\mathbf{b}_{ij} \tag{19}$$

In summary, $\partial\boldsymbol{\psi}_i^{\mathcal{M}}/\partial\alpha_j$ is 0 at the q_i-th entry, and other entries are provided by the equation above. The Jacobian of the simulated eigenvector at measured DOFs, $\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}\right) \in \mathbb{R}^{n_{\mathcal{M}}\times n_{\boldsymbol{\alpha}}}$ in Eq. (9), can be obtained as follows:

$$\begin{aligned}\mathbf{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}\right) &= \begin{bmatrix}\dfrac{\partial\boldsymbol{\psi}_i^{\mathcal{M}}}{\partial\alpha_1} & \dfrac{\partial\boldsymbol{\psi}_i^{\mathcal{M}}}{\partial\alpha_2} & \cdots & \dfrac{\partial\boldsymbol{\psi}_i^{\mathcal{M}}}{\partial\alpha_{n_{\boldsymbol{\alpha}}}}\end{bmatrix}\\ &= \left[\begin{array}{cccc}\partial\psi_{1,i}^{\mathcal{M}}/\partial\alpha_1 & \partial\psi_{1,i}^{\mathcal{M}}/\partial\alpha_2 & \cdots & \partial\psi_{1,i}^{\mathcal{M}}/\partial\alpha_{n_{\boldsymbol{\alpha}}}\\ \vdots & \vdots & \cdots & \vdots\\ \partial\psi_{q_i-1,i}^{\mathcal{M}}/\partial\alpha_1 & \partial\psi_{q_i-1,i}^{\mathcal{M}}/\partial\alpha_2 & \cdots & \partial\psi_{q_i-1,i}^{\mathcal{M}}/\partial\alpha_{n_{\boldsymbol{\alpha}}}\\ 0 & 0 & \cdots & 0\\ \psi_{q_i+1,i}^{\mathcal{M}}/\partial\alpha_1 & \partial\psi_{q_i+1,i}^{\mathcal{M}}/\partial\alpha_2 & \cdots & \partial\psi_{q_i+1,i}^{\mathcal{M}}/\partial\alpha_{n_{\boldsymbol{\alpha}}}\\ \vdots & \vdots & \cdots & \vdots\\ \psi_{n_{\mathcal{M}},i}^{\mathcal{M}}/\partial\alpha_1 & \psi_{n_{\mathcal{M}},i}^{\mathcal{M}}/\partial\alpha_2 & \cdots & \psi_{n_{\mathcal{M}},i}^{\mathcal{M}}/\partial\alpha_{n_{\boldsymbol{\alpha}}}\end{array}\right]\left|\begin{array}{c}\text{row \#}\\ 1\\ \vdots\\ q_i-1\\ q_i\\ q_i+1\\ \vdots\\ n_{\mathcal{M}}\end{array}\right.\end{aligned} \tag{20}$$

After obtaining the Jacobian of the simulated eigenvalue and eigenvector at measured DOFs, $\mathrm{D}_{\boldsymbol{\alpha}}(\lambda_i)$ and $\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_i^{\mathcal{M}}\right)$, the analytical Jacobian in Eq. (9) can be calculated.

3.2 Jacobian of modal property difference formulation #2: eigenvector difference approach

For the eigenvector difference approach in Eq. (5), the optimization variable $\mathbf{x}$ in Eq. (7) is the stiffness updating vector variable $\boldsymbol{\alpha}$. In order to derive the Jacobian of the eigenvector difference formulation in Eq. (5), a residual vector function $\mathbf{r}(\boldsymbol{\alpha})\colon \mathbb{R}^{n_{\boldsymbol{\alpha}}} \to \mathbb{R}^{n_{\mathcal{M}} \cdot n_{\text{modes}}}$ is defined as follows:

$$\mathbf{r}(\boldsymbol{\alpha}) = \begin{bmatrix} \mathbf{r}_1(\boldsymbol{\alpha}) \\ \vdots \\ \mathbf{r}_{n_{\text{modes}}}(\boldsymbol{\alpha}) \end{bmatrix} = \begin{bmatrix} \left(\lambda_1^{\text{EXP}} - \lambda_1(\boldsymbol{\alpha})\right)/\lambda_1^{\text{EXP}} \cdot w_{\lambda_1} \\ \left\{\boldsymbol{\psi}_{-q_1,1}^{\text{EXP},\mathcal{M}} - \boldsymbol{\psi}_{-q_1,1}^{\mathcal{M}}(\boldsymbol{\alpha})\right\} \cdot w_{\boldsymbol{\psi}_1} \\ \vdots \\ \left(\lambda_{n_{\text{modes}}}^{\text{EXP}} - \lambda_{n_{\text{modes}}}(\boldsymbol{\alpha})\right)/\lambda_{n_{\text{modes}}}^{\text{EXP}} \cdot w_{\lambda_{n_{\text{modes}}}} \\ \left\{\boldsymbol{\psi}_{-q_{n_{\text{modes}}},n_{\text{modes}}}^{\text{EXP},\mathcal{M}} - \boldsymbol{\psi}_{-q_{n_{\text{modes}}},n_{\text{modes}}}^{\mathcal{M}}(\boldsymbol{\alpha})\right\} \cdot w_{\boldsymbol{\psi}_{n_{\text{modes}}}} \end{bmatrix} \tag{21}$$

where $\mathbf{r}_i(\boldsymbol{\alpha})\colon \mathbb{R}^{n_{\boldsymbol{\alpha}}} \to \mathbb{R}^{n_{\mathcal{M}}}$ equals $\left[\left(\lambda_i^{\text{EXP}} - \lambda_i(\boldsymbol{\alpha})\right)/\lambda_i^{\text{EXP}} \cdot w_{\lambda_i};\quad \left\{\boldsymbol{\psi}_{-q_i,i}^{\text{EXP},\mathcal{M}} - \boldsymbol{\psi}_{-q_i,i}^{\mathcal{M}}(\boldsymbol{\alpha})\right\} \cdot w_{\boldsymbol{\psi}_i}\right], i = 1 \dots n_{\text{modes}}$. Using $\mathbf{r}(\boldsymbol{\alpha})$, the optimization problem in Eq. (5) for the eigenvector difference formulation can also be rewritten the same as Eq. (7), with the objective function $f(\boldsymbol{\alpha}) = \mathbf{r}(\boldsymbol{\alpha})^{\mathrm{T}}\mathbf{r}(\boldsymbol{\alpha})$. Again, the Jacobian for $f(\boldsymbol{\alpha})$, $\mathrm{D}_{\boldsymbol{\alpha}} f \in \mathbb{R}^{1 \times n_{\boldsymbol{\alpha}}}$ equals $\mathrm{D}_{\mathbf{r}} f \cdot \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}$ from the chain rule. However, the residual vector $\mathbf{r}$ has a different dimension for the MAC value formulation. For the eigenvector difference formulation, the first Jacobian term is $\mathrm{D}_{\mathbf{r}} f = 2\mathbf{r}^{\mathrm{T}} \in \mathbb{R}^{1 \times (n_{\mathcal{M}} \cdot n_{\text{modes}})}$. Meanwhile, the second term is $\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r} = [\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_1;\quad \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_2;\quad \cdots\quad \mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_{n_{\text{modes}}}] \in \mathbb{R}^{(n_{\mathcal{M}} \cdot n_{\text{modes}}) \times n_{\boldsymbol{\alpha}}}$, where each $\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i \in \mathbb{R}^{n_{\mathcal{M}} \times n_{\boldsymbol{\alpha}}}$ can be formed as follows:

$$\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i = \begin{bmatrix} -\dfrac{\mathrm{D}_{\boldsymbol{\alpha}}\left(\lambda_i(\boldsymbol{\alpha})\right)}{\lambda_i^{\text{EXP}}} \cdot w_{\lambda_i} \\ -\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_{-q_i,i}^{\mathcal{M}}(\boldsymbol{\alpha})\right) \cdot w_{\boldsymbol{\psi}_i} \end{bmatrix}, i = 1 \dots n_{\text{modes}} \tag{22}$$

The Jacobian of the *i*-th simulated eigenvalue and eigenvector at measured DOFs, $\mathrm{D}_{\boldsymbol{\alpha}}\left(\lambda_i(\boldsymbol{\alpha})\right)$ and $\mathrm{D}_{\boldsymbol{\alpha}}\left(\boldsymbol{\psi}_{-q_i,i}^{\mathcal{M}}(\boldsymbol{\alpha})\right)$, have been provided in Eq. (13) and Eq. (20).

3.3 Jacobian of modal dynamic residual formulation

To derive the Jacobian of the modal dynamic residual formulation in Eq. (6), define a vector variable $\mathbf{x} = \left[\boldsymbol{\alpha};\quad \boldsymbol{\psi}_1^{\mathcal{U}};\quad \cdots\quad \boldsymbol{\psi}_{n_{\text{modes}}}^{\mathcal{U}}\right] \in \mathbb{R}^{n_{\mathbf{x}}}$ with $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} + n_{\mathcal{U}} \cdot n_{\text{modes}}$ and a residual vector function $\mathbf{r}(\mathbf{x}) : \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}^{N \cdot n_{\text{modes}}}$ as:

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} \mathbf{r}_1(\mathbf{x}) \\ \vdots \\ \mathbf{r}_{n_{\text{modes}}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \left[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_1^{\text{EXP}}\mathbf{M}\right] \begin{Bmatrix} \boldsymbol{\psi}_1^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_1^{\mathcal{U}} \end{Bmatrix} \cdot w_1 \\ \vdots \\ \left[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_{n_{\text{modes}}}^{\text{EXP}}\mathbf{M}\right] \begin{Bmatrix} \boldsymbol{\psi}_{n_{\text{modes}}}^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_{n_{\text{modes}}}^{\mathcal{U}} \end{Bmatrix} \cdot w_{n_{\text{modes}}} \end{bmatrix} \tag{23}$$

where $\mathbf{r}_i(\mathbf{x}) : \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}^N$ equals $\left[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i^{\text{EXP}}\mathbf{M}\right] \begin{Bmatrix} \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_i^{\mathcal{U}} \end{Bmatrix} \cdot w_i, i = 1 \dots n_{\text{modes}}$. Again, the Jacobian for $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^{\mathrm{T}}\mathbf{r}(\mathbf{x})$, $\mathrm{D}_{\mathbf{x}}f \in \mathbb{R}^{1 \times n_{\mathbf{x}}}$ equals $\mathrm{D}_{\mathbf{r}}f \cdot \mathrm{D}_{\mathbf{x}}\mathbf{r}$ from the chain rule. The first Jacobian term is $\mathrm{D}_{\mathbf{r}}f = 2\mathbf{r}^{\mathrm{T}} \in \mathbb{R}^{1 \times (N \cdot n_{\text{modes}})}$ while the second term is $\mathrm{D}_{\mathbf{x}}\mathbf{r} = [\mathrm{D}_{\mathbf{x}}\mathbf{r}_1;\quad \mathrm{D}_{\mathbf{x}}\mathbf{r}_2;\quad \cdots\quad \mathrm{D}_{\mathbf{x}}\mathbf{r}_{n_{\text{modes}}}] \in \mathbb{R}^{(N \cdot n_{\text{modes}}) \times n_{\mathbf{x}}}$, where each $\mathrm{D}_{\mathbf{x}}\mathbf{r}_i \in \mathbb{R}^{N \times n_{\mathbf{x}}}$ can be expressed as follows:

$$\mathrm{D}_{\mathbf{x}}\mathbf{r}_i = \left[\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i \quad \mathrm{D}_{\boldsymbol{\psi}_1^{\mathcal{U}}}\mathbf{r}_i \quad \cdots \quad \mathrm{D}_{\boldsymbol{\psi}_{n_{\text{modes}}}^{\mathcal{U}}}\mathbf{r}_i\right], \qquad i = 1 \dots n_{\text{modes}} \tag{24}$$

In Eq. (24), $\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i \in \mathbb{R}^{N \times n_{\boldsymbol{\alpha}}}$ can be derived considering $\mathbf{K}(\boldsymbol{\alpha}) = \mathbf{K}_0 + \sum_{j=1}^{n_{\boldsymbol{\alpha}}} \alpha_j \mathbf{K}_j$ from Eq. (1):

$$\mathrm{D}_{\boldsymbol{\alpha}}\mathbf{r}_i = \left[\mathbf{K}_1 \begin{Bmatrix} \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_i^{\mathcal{U}} \end{Bmatrix} \cdot w_i \quad \mathbf{K}_2 \begin{Bmatrix} \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_i^{\mathcal{U}} \end{Bmatrix} \cdot w_i \quad \cdots \quad \mathbf{K}_{n_{\boldsymbol{\alpha}}} \begin{Bmatrix} \boldsymbol{\psi}_i^{\text{EXP},\mathcal{M}} \\ \boldsymbol{\psi}_i^{\mathcal{U}} \end{Bmatrix} \cdot w_i\right], \qquad i = 1 \dots n_{\text{modes}} \tag{25}$$

and $\mathrm{D}_{\boldsymbol{\psi}_j^{\mathcal{U}}}\mathbf{r}_i \in \mathbb{R}^{N \times n_{\mathcal{U}}}, j = 1 \dots n_{\text{modes}}$ in Eq. (24) can be expressed as:

$$\mathrm{D}_{\boldsymbol{\psi}_j^{\mathcal{U}}}\mathbf{r}_i = \begin{cases} \left[\mathbf{K}(\boldsymbol{\alpha}) - \lambda_i^{\text{EXP}}\mathbf{M}\right] \begin{Bmatrix} \mathbf{0}_{n_{\mathcal{M}} \times n_{\mathcal{U}}} \\ \mathbf{I}_{n_{\mathcal{U}}} \end{Bmatrix} \cdot w_i \text{ for } j = i \\ \mathbf{0}_{N \times n_{\mathcal{U}}} \text{ for } j \neq i \end{cases} \tag{26}$$

where $\mathbf{0}_{n_{\mathcal{M}} \times n_{\mathcal{U}}}$ and $\mathbf{0}_{N \times n_{\mathcal{U}}}$ denote zero matrices and $\mathbf{I}_{n_{\mathcal{U}}}$ denotes the identity matrix, respectively.

4 Introduction of SMU package

Figure 1 shows the overall flowchart of the SMU package for structural model updating (Otsuki et al., 2021b). The GitHub package contains MATLAB subroutines for model updating with example codes for several structures. The main function `StructModelUpdating` performs a single run of structural model updating using one of the three

formulations selected by users. Analytical gradients for all three formulations are implemented in SMU. In order to increase the chance of finding a better local minimum for these nonconvex optimization problems, randomly generated starting points can be assigned by `MultiRunModelUpdating`. This script calls `StructModelUpdating` to conduct multiple runs of model updating, i.e. performs gradient searches from all the random starting points. To solve each corresponding optimization problem from one assigned starting point in the feasible region of variables, SMU uses the optimization solvers `lsqnonlin` and `fmincon` available in the MATLAB optimization toolbox (MathWorks Inc., 2019). The GitHub package provides a number of structural examples, currently including a 4-DOF shear model, an 18-DOF shear model, a steel pedestrian bridge, and a concrete building frame. Model updating is performed for structures using both simulation and experiment data. Additionally, when users want to perform model updating of their structure using the SMU package, users only need to provide the mass and stiffness information of the initial FE model of the structure along with experimentally obtained eigenvalues and eigenvectors for a certain number of modes.

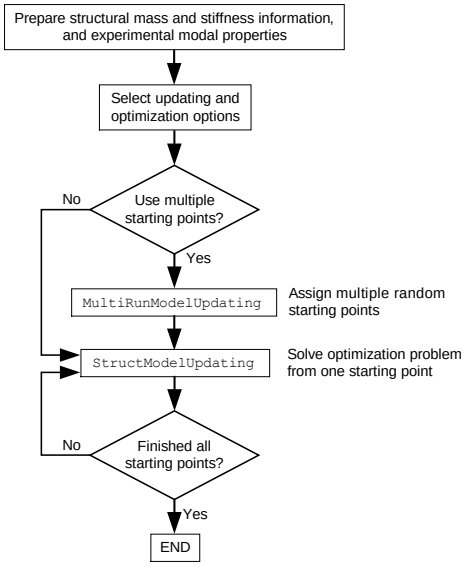


Figure 1. Overall flowchart of SMU

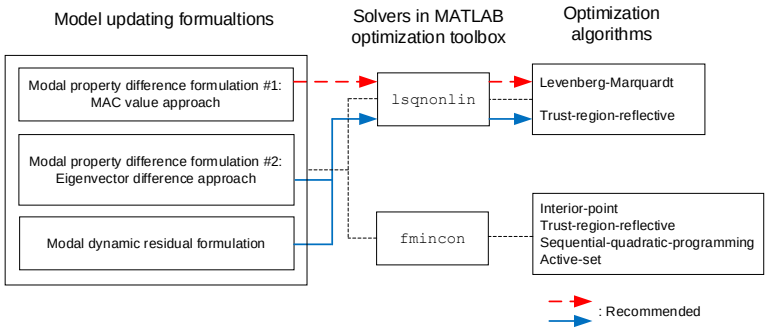


Figure 2. Recommended MATLAB solvers and optimization algorithms for each formulation

Figure 2 shows the recommended optimization solver and algorithm for the model updating formulations. Note that the computational time of the FE model updating depends not only on the selection of optimization solvers and algorithms but also the size of the optimization problem, the use of numerical or analytical Jacobian, and the selection of formulations, which will be illustrated in two example structures in Section 5 and 6. When minimizing

least-squares objective functions, `lsqnonlin` is computationally much more efficient than `fmincon` and thus recommended for a first try. The `lsqnonlin` solver minimizes the objective function $f(\mathbf{x}) = \mathbf{r}(\mathbf{x})^{\mathrm{T}}\mathbf{r}(\mathbf{x}) = \|\mathbf{r}(\mathbf{x})\|_2^2$ in Eq. (7), i.e. the square of the $\mathcal{L}_2$-norm of a residual vector function $\mathbf{r}(\mathbf{x})$. The gradient, $\nabla f(\mathbf{x}) \in \mathbb{R}^{n_\mathbf{x}}$ and Hessian, $\nabla^2 f(\mathbf{x}) \in \mathbb{R}^{n_\mathbf{x} \times n_\mathbf{x}}$ of $f(\mathbf{x})$: $\mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}$ can be expressed as follows:

$$\nabla f(\mathbf{x}) = 2[\mathrm{D}_\mathbf{x}\mathbf{r}]^{\mathrm{T}} \cdot \mathbf{r}(\mathbf{x}) \tag{27}$$

$$\nabla^2 f(\mathbf{x}) = 2[\mathrm{D}_\mathbf{x}\mathbf{r}]^{\mathrm{T}} \cdot \mathrm{D}_\mathbf{x}\mathbf{r} + 2\sum_{i=1}^{n_\mathbf{r}} r_i(\mathbf{x})\,\nabla^2 r_i \tag{28}$$

where $\mathrm{D}_\mathbf{x}\mathbf{r} \in \mathbb{R}^{n_\mathbf{r} \times n_\mathbf{x}}$ is defined as the Jacobian matrix of the scalar residuals $(r_i, i = 1 \cdots n_\mathbf{r})$ with respect to the optimization variables $(x_j, j = 1 \cdots n_\mathbf{x})$. Neglecting the higher-order second term in $\nabla^2 f(\mathbf{x})$, the optimization algorithms adopted by `lsqnonlin` in MATLAB uses $2[\mathrm{D}_\mathbf{x}\mathbf{r}]^{\mathrm{T}} \cdot \mathrm{D}_\mathbf{x}\mathbf{r}$ to approximate the Hessian matrix. At each step of the optimization process, by default, `lsqnonlin` numerically calculates the search gradient, $\nabla f(\mathbf{x})$, of the objective function using the finite forward difference method (LeVeque, 2007). The numerically calculated gradient results are affected by the difference $\Delta\mathbf{x}$, i.e. the step size of $\mathbf{x}$, and are more prone to inaccuracies. Meanwhile, instead of using the numerically calculated gradient, `lsqnonlin` also accepts a user-provided analytical formulation of the gradient. Given that the gradient simply equals the transpose of the Jacobian, i.e. $\nabla f(\mathbf{x}) = (\mathrm{D}_\mathbf{x} f)^{\mathrm{T}}$, the analytical gradient for each formulation has been derived in Section 3 and implemented in SMU.

At each iteration, the optimization solver evaluates the objective function value once or several times at points near the current value of $\mathbf{x}$, which is called the number of function evaluations or `F-count` in the MATLAB optimization toolbox (MathWorks Inc., 2019). When using the default numerical gradient, i.e., finite forward difference method, each iteration needs to evaluate the objective function $n_\mathbf{x} + 1$ times in the neighbor of the current $\mathbf{x}$. On the other hand, when the formulation of the analytical gradient is provided, the function evaluation happens only once at each iteration. Therefore, when the computation of the analytical gradient itself is not expensive, the analytical gradient can not only provide better accuracy, but also require less computational time due to the smaller number of function evaluations. The advantage of using the analytical gradient is more significant when $n_\mathbf{x}$ is large, i.e., a large-scale optimization problem.

Two optimization algorithms available in `lsqnonlin` are the Levenberg-Marquardt (L-M) algorithm (Moré, 1978) and the trust-region-reflective (TRR) algorithm (Coleman and Li, 1996). The L-M algorithm is a combination of the steepest decent and Gauss-Newton algorithms and is specially designed to solve nonlinear least-squares problems. At every iteration, the algorithm first linearizes the objective function with respect to the corresponding optimization variables. When the current solution is far from a local optimum, the L-M algorithm approaches the steepest descent algorithm. On the other hand, when the current solution is close to a local optimum, the L-M algorithm approaches the Gauss-Newton algorithm. The TRR algorithm approximates the original problem with a quadratic subproblem within a small region around the current solution point, i.e. a trusted region. The quadratic subproblem is formulated using the same gradient and approximated Hessian of the original problem. By solving the quadratic subproblem using the two-dimensional subspace approach, a solution of the current subproblem can be obtained (Byrd et al., 1988; Branch et al., 1999). If the decrease of the objective function evaluated at the current step is within the prescribed upper and lower bounds, the solution will be accepted, and the algorithm will continue with the next iteration. Otherwise, the trusted region at the current iteration will be adjusted, and the quadratic subproblem is solved again with the new region. Iteratively, the optimization converges to a local minimum of the objective function.

The L-M algorithm in the current version of SMU does not allow to set upper/lower bounds of optimization variables. This can be a drawback in model updating using experimental data because appropriate upper/lower bounds ensure a physically meaningful optimal solution. The TRR algorithm in MATLAB does allow setting the upper/lower bounds of optimization variables; however, the TRR algorithm is not applicable for underdetermined problems, i.e. when $n_{\mathbf{r}} < n_{\mathbf{x}}$. The MAC value approach for model updating (Eq. (3)) oftentimes provides an underdetermined problem. For these reasons, we recommend to first try the L-M algorithm for the MAC value approach. Whereas for the eigenvector difference approach (Eq. (5)) and the modal dynamic residual formulation (Eq. (6)), we recommend to first try the TRR algorithm.

All the model updating formulations (Eqs. (3), (5), and (6)) in general provide non-convex optimization problems. If the optimization problem is nonconvex, none of the existing algorithms in the MATLAB optimization toolbox can guarantee the global optimality. Therefore, besides starting the search from random initial points in the feasible region (`MultiRunModelUpdating`), it is also recommended to use different optimization algorithms and compare the model updating results. Notice that the optimization algorithms available in `lsqnonlin` and `fmincon` are different. Users may select `fmincon` when there is a particular interest in one of the algorithms available in

`fmincon`, such as the interior-point method, the sequential-quadratic-programming algorithm, and the active-set method (MathWorks Inc., 2019).

5 Example 1: 18-story steel frame

5.1 Overview of the 18-story steel frame example

This section demonstrates an application example of SMU, using a numerical model based on a one-third scale 18-story steel frame. The physical structure was constructed and tested at the E-Defense in 2014 (Figure 3(a)) (Suita et al., 2015). The structure's behavior represents that of typical steel high-rise buildings constructed in the 1980s to 1990s in Japan. The plane dimension of each floor is 5×6 m, and the total height is 25.35 m. In this numerical study, the overall structure is simplified to an 18-DOF shear model as shown in Figure 3(b).

(a) Front view (b) 18-DOF model

Figure 3. 18-story steel frame

The structural properties for this numerical study are presented in Table 1. It is assumed that the floor masses are accurate and do not require updating. The initial stiffness values are calculated from nominal material properties. The "actual" structure in this simulation corresponds to the assigned stiffness updating variable α_j^{act} for each story, which is to be identified through model updating. The value of each updating variable represents the relative change of a stiffness parameter from its initial/nominal value. The number of measured DOFs is assumed to be seven, corresponding to the 1st, 4th, 7th, 9th, 12th, 15th, and 18th DOFs. Therefore, $n_{\mathcal{M}}$ = 7 and $n_{\mathcal{U}}$ = 11. The number of

"experimental" modes available in this model updating is set as four ($n_{\text{modes}} = 4$). For the MAC value and the eigenvector difference approaches, the number of optimization variables is $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} = 18$. For the modal dynamic residual formulation, the number is $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} + n_U \times n_{\text{modes}}$ = 18+11×4 = 62.

Table 1. Structural properties of the 18-DOF model for this numerical study

Floor	Weight (kN)	Initial inter-story stiffness (kN/m)	Actual inter-story stiffness (kN/m)	Updating variable	Actual updating variable α_j^{act}
18*	202	36,300	43,560	α_{18}	0.20
17	206	49,100	58,920	α_{17}	0.20
16	206	56,200	61,820	α_{16}	0.10
15*	206	61,900	52,615	α_{15}	-0.15
14	206	66,000	69,300	α_{14}	0.05
13	206	71,200	60,520	α_{13}	-0.15
12*	206	78,800	98,500	α_{12}	0.25
11	208	82,400	107,120	α_{11}	0.30
10	208	84,000	100,800	α_{10}	0.20
9*	208	87,600	78,840	α_9	-0.10
8	208	93,800	117,250	α_8	0.25
7*	208	96,300	110,745	α_7	0.15
6	208	99,000	84,150	α_6	-0.15
5	208	102,800	113,080	α_5	0.10
4*	208	102,800	92,520	α_4	-0.10
3	208	107,300	101,935	α_3	-0.05
2	208	109,200	114,660	α_2	0.05
1*	208	115,500	121,275	α_1	0.05
Note: (* measured/instrumented DOFs)					

5.2 Procedure illustration for structural model updating using SMU

This section demonstrates the procedures for updating the 18-DOF model. The SMU MATLAB package can be downloaded from GitHub (Wang et al., 2019b). The folder "Structural-Model-Updating\SMU" contains shared MATLAB subroutines for structural model updating. This folder should be added into the MATLAB path first. The MATLAB code that contains structural properties of this 18-DOF simulation study can be found in "Structural-Model-Updating\Examples\EighteenStoryStructure\Simulation". Figure 4 shows a part of the main script for model updating of the 18-DOF model. The script at first sets the basic parameters of the structure. As mentioned, this example assumes four "experimental" modes ($n_{\text{modes}} = 4$) are available; the measured DOFs are identified. These parameters can be

easily changed by users to see the effect on model updating results. The mass matrix $\mathbf{M}$, initial stiffness matrix $\mathbf{K}_0$, and influence matrix $\mathbf{K}_j$ in Eq. (1) are constructed based on properties in Table 1. Upon construction or loading from a data file, they are assembled in a MATLAB structure array `structModel`. Experimental eigenvalues λ_i^{EXP} and eigenvectors $\boldsymbol{\psi}_i^{\mathrm{EXP},\mathcal{M}}$ used in the model updating formulations – see Eq. (3), Eq.(5), and Eq. (6) – are then generated using the "actual" values of the stiffness updating variables as listed in Table 1, and they are assigned to variables

```
%% Basic parameters
N = 18; % # of DOFs of the whole structure
n_modes = 4; % # of measured experimental modes
measDOFs = [1;4;7;9;12;15;18];  % Measured DOFs
```

⋮

```
%% Optimization settings
optimzOpts.toolBox = 'lsqnonlin';                  % Optimization solver
optimzOpts.optAlgorithm = ' Levenberg-Marquardt'; % Optimization algorithm
optimzOpts.gradSel = 'on';                         % on = analytical gradient; off = numerical
                                                   % gradient

%% Model updating settings
updatingOpts.formID = 1;                           % 1: Modal property difference formulation with
                                                   % MAC values
                                                   % 2: Modal property diff. formulation with
                                                   % eigenvectors
                                                   % 3: Modal dynamic residual formulation
modeIndex = 1 : n_modes;                          % Indexes of simulated modes for matching exp. modes
updatingOpts.modeMatch = 2;                        % 1: Without forced matching
                                                   % 2: With forced matching
updatingOpts.simModesForExpMatch = modeIndex;
if (updatingOpts.formID < 3.0)
    % Optimizaiton variable bounds for modal property difference formulations
    updatingOpts.x_lb = -ones(n_alpha, 1);
    updatingOpts.x_ub =  ones(n_alpha, 1);
else
    % Optimizaiton variable bounds for modal dynamic residual formulations
    updatingOpts.x_lb = [-ones(n_alpha, 1); -2 * ones(num_unmeasDOFs * n_modes, 1)];
    updatingOpts.x_ub = [ones(n_alpha, 1);  2 * ones(num_unmeasDOFs * n_modes, 1)];
end

% Weighting factors
expModes.lambdaWeights = ones(n_modes, 1);
expModes.psiWeights = ones(n_modes, 1);
expModes.resWeights = ones(n_modes, 1);

%% MultiStart optimization
numRuns = 100;
randSeed = 5;
filename = ['EighteenStoryFrame_form' num2str(updatingOpts.formID) '_JAC' ...
            optimzOpts.gradSel '_' optimzOpts.optAlgorithm '.mat'];

MultiRunModelUpdating
```

Figure 4. Main script for model updating of the 18-DOF model in simulation

`lambdaExp` and `psiExp_m`, containing λ_i^{EXP} and $\boldsymbol{\psi}_i^{\mathrm{EXP},\mathcal{M}}$, respectively. The script next calls the SMU subroutines to perform structural model updating. Before calling the subroutines, options for optimization algorithms can be specified by users. The example below uses `lsqnonlin` and the L-M algorithm with analytical gradient.

The `formID = 1` means that the modal property difference formulation with the MAC value approach in Eq. (3) is used. For the matching method between simulated and experimental modes, `modeMatch` and `modeIndex` are assigned. When `modeMatch` is set as `1`, at every iteration step, the program goes through remaining simulated modes specified by `modeIndex` and pairs each experimental mode with a remaining simulated mode with the largest MAC value; the identified pairs of $\boldsymbol{\psi}_i^{\mathrm{EXP},\mathcal{M}}$ and $\boldsymbol{\psi}_i^{\mathcal{M}}$ are then used to evaluate the objective function and search gradient. When `modeMatch` is set as `2`, the program strictly matches the first experimental mode with the first simulated mode specified by `modeIndex`, the second experimental mode with the second one in `modeIndex`, *etc*. The `modeMatch = 2` setting should be used only when users are confident that all the lowest few modes are captured by experimental data, i.e. there is no missing or unmeasured/undetected mode from the experimental data.

In this example, the `formID = 1` is selected to use the MAC value formulation in Eq. (3). The upper and lower bounds (`x_lb` and `x_ub`) are set as ± 1 for the stiffness updating variables $\boldsymbol{\alpha}$. In case the modal dynamic residual formulation in Eq. (6) is used (`formID = 3`), the upper and lower bounds are set as ± 1 for the stiffness updating variables $\boldsymbol{\alpha}$ and ± 2 for the unmeasured entries of eigenvectors $\boldsymbol{\psi}_i^{\mathcal{U}}$. Note that the eigenvectors in simulation and experiment should have been normalized so that the largest entry of $\boldsymbol{\psi}_i^{\mathrm{EXP},\mathcal{M}}$ equals 1. In this demonstration, the weighting factors are all equal to 1 and are assigned to the corresponding field of the structure array `expModes`, including w_{λ_i}, $w_{\boldsymbol{\psi}_i}$, and w_i. In practice, depending on the confidence in experimental data and the knowledge of the structure's dynamics, appropriate weighting factors should be specified. Next, the options for `MultiRunModelUpdating` are specified. The 100 randomly generated starting points in the feasible regions are used. The variable `randSeed` is the random seed value for the MATLAB function "`rng`". The variable `filename` will be used for the result file name (.mat file) that contains model updating results for all the starting points. Upon running the code, the iteration history will be shown in the MATLAB Command Window, and the result file (.mat file) will be automatically generated. For this study, we use a PC with an Intel i7-7700 CPU and 16 GB RAM.

5.3 Model updating results: MAC value approach

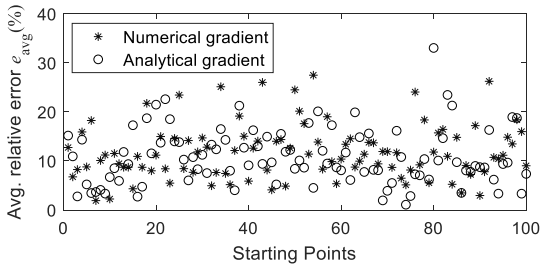
For this simulation, the MAC value approach in Eq. (3) provides $n_{\mathbf{r}} = 2 \cdot n_{\text{modes}} = 2 \cdot 4 = 8$ and $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} = 18$. Hence, $n_{\mathbf{r}} < n_{\mathbf{x}}$, i.e. the optimization problem is underdetermined; the TRR algorithm is not applicable, and the L-M algorithm is used to solve the optimization problem. When using MATLAB `lsqnonlin` with the L-M algorithm in the current version of SMU, the optimal result sets that are out of the assigned bounds are rejected and the corresponding starting point is replaced with the next randomly generated point that achieves valid optimal results. For comparison, both numerical and analytical gradients are used. For each optimum parameter set $\boldsymbol{\alpha}^*$, the relative error of the stiffness updating variables is calculated as:

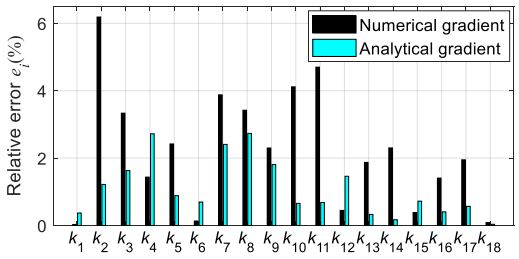
$$e_i = \frac{|\alpha_i^* - \alpha_i^{\text{act}}|}{1 + \alpha_i^{\text{act}}} \times 100\%, \qquad i = 1 \cdots n_{\boldsymbol{\alpha}} \tag{29}$$

where α_i^* is the *i*-th entry of $\boldsymbol{\alpha}^*$ and α_i^{act} is the actual value as listed in Table 1. Then, the average relative error e_{avg} is calculated for this optimum parameter set $\boldsymbol{\alpha}^*$ as:

$$e_{\text{avg}} = \frac{1}{n_{\boldsymbol{\alpha}}} \sum_{i=1}^{n_{\boldsymbol{\alpha}}} e_i \tag{30}$$

Figure 5(a) plots the e_{avg} of the optimum $\boldsymbol{\alpha}^*$ from all of the 100 starting points. In this plot, many of the optimization searches end with a solution that has a large relative error e_{avg}; some relative errors are close to or even above 30%. The reason is that in this simulation, the maximum number of search iterations from each starting point is set as the default value of 400 by SMU. If desired, the number can be easily increased for better accuracy while consuming longer runtime. Among 100 sets of the optimum vector variable $\boldsymbol{\alpha}^*$, the best parameter set is chosen as the one corresponding to the minimum objective function value. Figure 5(b) shows the relative error of this best parameter set. Both numerical and analytical gradients achieve a solution close to the actual value of updating variables, but the analytical gradient obtains better accuracy. The total computational time is 475.8 seconds when using the numerical gradient while it is 386.9 seconds for the analytical gradient.



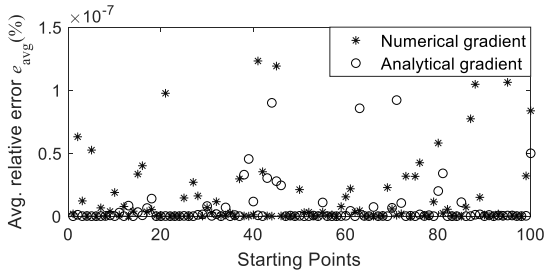


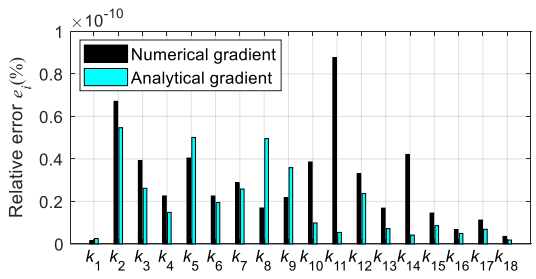
(a) Average relative errors of updating variables (b) Relative errors of updated stiffnesses

Figure 5. Model updating results of 18-DOF model by MAC value approach using L-M algorithm with numerical and analytical gradients

5.4 Model updating results: Eigenvector difference approach

The eigenvector difference approach in Eq. (5) provides $n_{\mathbf{r}} = n_{\mathcal{M}} \cdot n_{\text{modes}} = 7 \cdot 4 = 28$ and $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} = 18$. Hence, $n_{\mathbf{r}} > n_{\mathbf{x}}$, i.e. the optimization problem is determined. Therefore, the TRR algorithm is applicable and used as recommended in Figure 2. Both numerical and analytical gradients are studied for comparison. Figure 6(a) shows the average relative error for each starting point. Figure 6(b) plots the relative error of the stiffness values for the minimum objective function value among 100 optimal sets. It can be concluded that for this relatively simple example, the eigenvector difference approach, using both numerical and analytical gradients, is able to identify accurate stiffness values of the model. The total computational time is 49.2 seconds when using the numerical gradient and 23.6 seconds for the analytical gradient. Compared to the MAC value approach, the eigenvector difference approach achieves higher accuracy and much better computational efficiency.



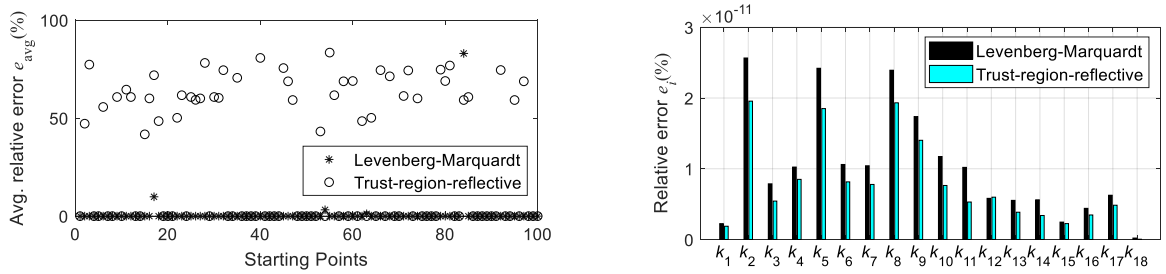


(a) Average relative errors of updating variables (b) Relative errors of updated stiffnesses

Figure 6. Model updating results of 18-DOF model by eigenvector difference approach using TRR algorithm with numerical and analytical gradients

5.5 Model updating results: Modal dynamic residual formulation

The modal dynamic residual formulation in Eq. (6) provides $n_{\mathbf{r}} = n_N \cdot n_{\text{modes}} = 18 \cdot 4 = 72$ and $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} + n_{\mathcal{U}} \cdot n_{\text{modes}} = 18 + 11 \cdot 4 = 62$. Hence, $n_{\mathbf{r}} > n_{\mathbf{x}}$, i.e. the optimization problem is determined. To compare the different optimization algorithms, the L-M algorithm and the TRR algorithm are both used with the analytical gradient. Figure 7(a) shows the average relative error of the updated result from each starting point. Compared to the L-M algorithm, several optimization searches using the TRR algorithm fail to converge at the correct value. Figure 7(b) plots the best solution among 100 optimal sets. The accuracy of the updated results is better than the MAC value approach and nearly the same as the eigenvector difference approach. The total computational time is 265.0 seconds for the L-M algorithm and 45.3 seconds for the TRR algorithm.



(a) Average relative errors of updating variables (b) Relative errors of updated stiffnesses

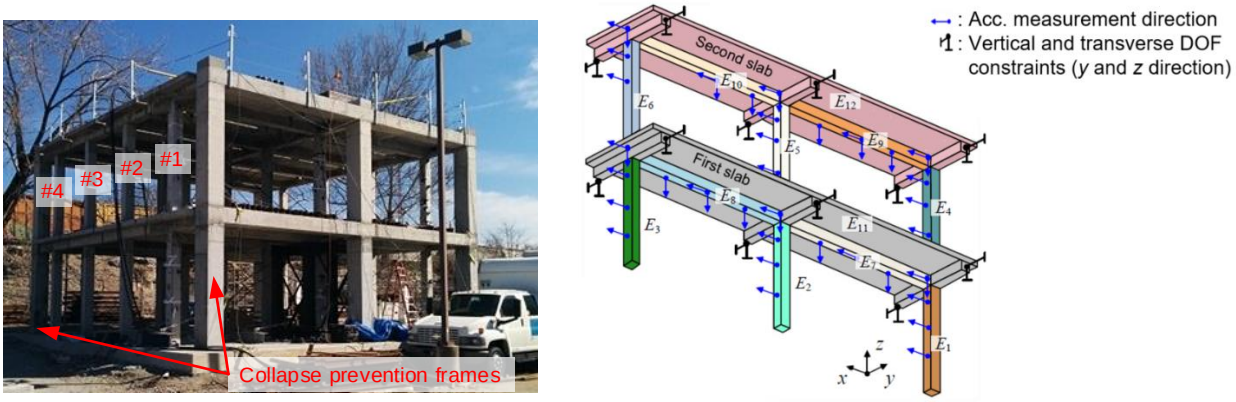
Figure 7. Model updating results of 18-DOF model by modal dynamic residual formulation using L-M and TRR algorithm with analytical gradient

In general, with eigenvectors $\boldsymbol{\psi}_i^{\mathcal{U}}$ among the optimization variables, the modal dynamic residual formulation can be computationally expensive when the number of unmeasured DOFs is large. In addition, as aforementioned, because the formulation does not directly minimize the discrepancy of modal properties, the updated model may provide modal properties that are more noticeably different from experimental data. Our overall experience is that the modal dynamic residual formulation usually does not perform well with larger-scale experimental data. Therefore, while for completeness the formulation is investigated in this section, it will not be studied further in the next section.

6 Example 2: concrete building frame

6.1 Overview of the concrete building frame example

The second structure studied in this research is a concrete building frame (Figure 8(a)), which simulates a full-scale test structure in the Structural Engineering and Materials Laboratory on the Georgia Tech campus. The test frame structure is representative of low-rise reinforced concrete office buildings in the central and eastern United States built in the 1950s-1970s (Dong et al., 2016). The structure consists of a set of four identical two-story two-bay concrete frames. Each frame was constructed with a gap from its neighboring frames, allowing free in-plane longitudinal movement and can thus be modeled independently from the other frames. Frame #1 is used in this study. The columns and beams for frame #1 are modeled with frame elements in SAP2000 (Figure 8(b)).



(a) Front view of the four testing frames, enveloped by two collapse-prevention frames

(b) Model of the 2-story 2-bay concrete building frame and sensor instrumentation

Figure 8. Concrete building frame (height in *z*: 2 × 12 ft.; length in *x*: 2 ×18 ft.; width in *y*: 9 ft.)

Corresponding to dense sensor instrumentation, seven segments are allocated per column on each story, and twelve segments per beam in each bay. In SAP2000 to ensure a stiffness contribution from both concrete and rebar, along every column or beam segment one frame element is assigned for the concrete material and another frame element is assigned for the steel reinforcement. Each floor slab is meshed into 175 shell elements. In total, the FE model of the concrete building frame has 2,302 DOFs. The modeling software SAP2000 assigns non-zero

concentrated mass only to the translational DOFs. As a result, the mass matrix (**M**) is a diagonal matrix whose diagonal entries associated with rotational DOFs equal zero.

Figure 8(b) also shows the accelerometer instrumentation for this simulation study, and the corresponding measurement directions. A total of 43 DOFs are measured, i.e. the length of $\boldsymbol{\psi}_i^{\mathrm{EXP},\mathcal{M}}$ and $\boldsymbol{\psi}_i^{\mathcal{M}}(\boldsymbol{\alpha})$ is $n_{\mathcal{M}} = 43$. The accelerometers measure longitudinal and vertical vibration, i.e. *x* and *z* directions. Thus, only in-plane vibration mode shapes, i.e. in the *x*-*z* plane, can be extracted from measurement data. To avoid the side effect of out-of-plane mode shapes (in the *y*-*z* plane) on the FE model updating, the vertical and transverse DOFs (*y* and *z* direction) at both ends of the three transverse beams (along the *y* direction) on each slab are constrained. Lastly, at the bottom of the three columns below the first slab, all six DOFs are constrained to represent an ideal fixed support.

This study updates twelve elastic moduli of concrete members, denoted as $E_1 \sim E_{12}$ in Figure 8(b). In the first story, $E_1 \sim E_3$ represent the concrete elastic moduli of members in the three columns; E_7 and E_8 represent the concrete elastic moduli of longitudinal beam members (along the *x* direction); and E_{11} represents the concrete elastic moduli of the first slab and the associated transverse beam members (along the *y* direction). Similarly, other moduli for the second story can be found in Figure 8(b). While this study only involves simulation, the selection of moduli corresponds to different concrete pours during the construction, and thus is in preparation for future model updating of the as-built structure with experimental data. Compared to concrete, the estimated elastic modulus of steel reinforcement is considered to be sufficiently accurate; therefore, it is not being updated in this study.

For all the concrete moduli being updated, Table 2 lists the nominal and actual values. In total, there are 12 updating variables for this model updating, i.e. $n_{\boldsymbol{\alpha}} = 12$. The column α_i^{act} in Table 2 lists the actual values of $\boldsymbol{\alpha}$, i.e. the ideal solutions to be identified through FE model updating. For the model updating of this concrete structure this paper presents the results of the modal property difference formulations with the MAC value approach (Eq. (3)) and the eigenvector difference approach (Eq. (5)). It is assumed that the first three vibration modes ($n_{\mathrm{modes}} = 3$) are available. The upper and lower bounds of $\boldsymbol{\alpha}$ are set to be 1 and -1, respectively. The weightings are set as $w_{\lambda_i} = 1$ and $w_{\boldsymbol{\psi}_i} = 1$, respectively. The optimization process is initiated from 100 random starting points within the bounds of $\boldsymbol{\alpha}$. The maximum number of iterations is set as 1000. Same as the simulation for the 18-DOF model, a PC with an Intel i7-7700 CPU and 16 GB RAM is used.

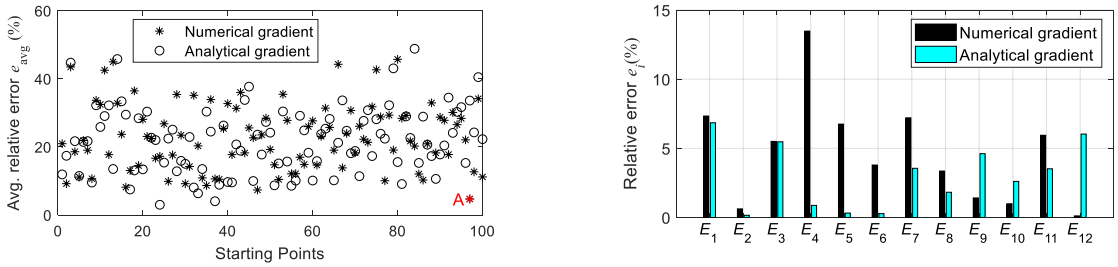
Table 2. Structural properties of the concrete building frame model for this numerical study

Stiffness parameters		Nominal value (N/mm^2)	Actual value (N/mm^2)	Updating variables	Actual updating variable α_j^{act}
Elastic moduli of concrete members	E_1	26890	24201	α_1	-0.10
	E_2	25511	30613	α_2	0.20
	E_3	25511	30613	α_3	0.20
	E_4	22063	20960	α_4	-0.05
	E_5	22063	26476	α_5	0.20
	E_6	22063	25373	α_6	0.15
	E_7	22063	25373	α_7	0.15
	E_8	22063	24270	α_8	0.10
	E_9	23442	21098	α_9	-0.10
	E_{10}	23442	19926	α_{10}	-0.15
	E_{11}	22063	26476	α_{11}	0.20
	E_{12}	23442	26958	α_{12}	0.15

6.2 Model updating results: MAC value approach

The MAC value approach (Eq. (3)) in this simulation provides $n_{\mathbf{r}} = 2 \cdot n_{\text{modes}} = 2 \cdot 3 = 6$ while $n_{\mathbf{x}} = n_{\boldsymbol{\alpha}} = 12$. As a result, $n_{\mathbf{r}} < n_{\mathbf{x}}$ and the problem is underdetermined; the TRR algorithm is not applicable and the L-M algorithm is used. Recall that the L-M algorithm in the current version of SMU does not allow upper/lower bounds. Therefore, when the converged optimal solution is out of the range [-1,1], the solution was discarded, and a new starting point was assigned. For comparison, both numerical and analytical gradients are used during optimization searches. For each of the 100 successful runs, Figure 9(a) displays the average relative error e_{avg} (Eq. (30)) for all $n_{\boldsymbol{\alpha}} = 12$ stiffness updating variables after discarding ten optimal result sets that are out of the [-1,1] bounds during the optimization process. Similar to the case for the MAC value approach of the 18-DOF model, a large number of optimization searches end with a large relative error e_{avg}. For the solution set that achieves the minimum objective function value among the 100 starting points, Figure 9(b) plots the relative error e_i of each stiffness parameter. The figure shows that the obtained stiffness parameter values are not reasonable, with the maximum relative error larger than 13% using the numerical gradient and 6.5% using the analytical gradient. It took 18 hours and 58 minutes to obtain the 100 optimal result sets that are within the bounds when using the numerical gradient, while it took 22 hours and 58 minutes when using the analytical gradient. While the analytical gradient provides more accurate solution, due to the computation cost of the analytical gradient for the MAC value approach, the numerical gradient required less computational time than the analytical gradient in this example. In conclusion, for this concrete frame structure, the MAC value approach

cannot provide reasonable FE model updating results using either the numerical gradient or analytical gradient during the optimization process.



(a) Average relative errors of updating variables

(b) Relative errors of updated stiffness parameters

Figure 9. Model updating results for the concrete building frame by MAC value approach using the L-M algorithm with numerical and analytical gradients

Further study on the nonconvexity of the MAC value approach is conducted, using solution A in Figure 9(a). The average relative error of solution A is 4.71% and the objective function equals 4.71×10^{-10}. While it is impossible to visualize $f(\boldsymbol{\alpha})$ in 12-dimensional space, Figure 10 displays a walk from solution A, denoted as $\boldsymbol{\alpha}_{\mathrm{A}}$, to the actual value $\boldsymbol{\alpha}^{\mathrm{act}}$ shown in Table 2, along a hyperline in $\mathbb{R}^{12}$ space. The hyperline is defined with a scalar θ; along the hyperline, the values of updating variables are $\boldsymbol{\alpha}(\theta) = \boldsymbol{\alpha}_{\mathrm{A}} + \theta \cdot (\boldsymbol{\alpha}^{\mathrm{act}} - \boldsymbol{\alpha}_{\mathrm{A}})$. When $\theta = 0, \boldsymbol{\alpha} = \boldsymbol{\alpha}_{\mathrm{A}}$; when $\theta = 1$, $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\mathrm{act}}$. Accordingly, the *y*-axis represents the MAC value objective function (Eq. (3)) evaluated at different values of $\boldsymbol{\alpha}$ along the hyperline:

$$f\big(\boldsymbol{\alpha}(\theta)\big) = f\big(\boldsymbol{\alpha}_{\mathrm{A}} + \theta \cdot (\boldsymbol{\alpha}^{\mathrm{act}} - \boldsymbol{\alpha}_{\mathrm{A}})\big) \tag{31}$$

Had $f(\boldsymbol{\alpha})$ been convex on $\boldsymbol{\alpha}$, $f\big(\boldsymbol{\alpha}(\theta)\big)$ should also be convex on θ (see page 68 in the 7th Ed. of reference (Boyd and Vandenberghe, 2004)). Figure 10 clearly shows that there are two valleys along the hyperline, located at $\theta = 0$ and 1, respectively. Therefore, the optimization problem in Eq. (3) for the MAC value approach is confirmed to be non-convex, and off-the-shelf local optimization algorithms cannot ensure global optimality.

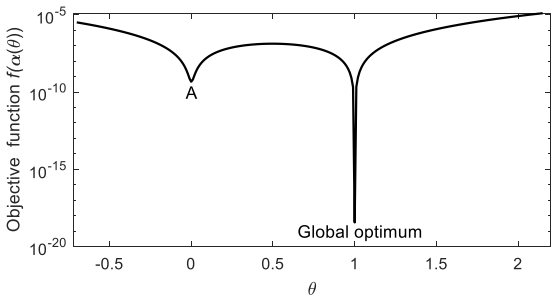
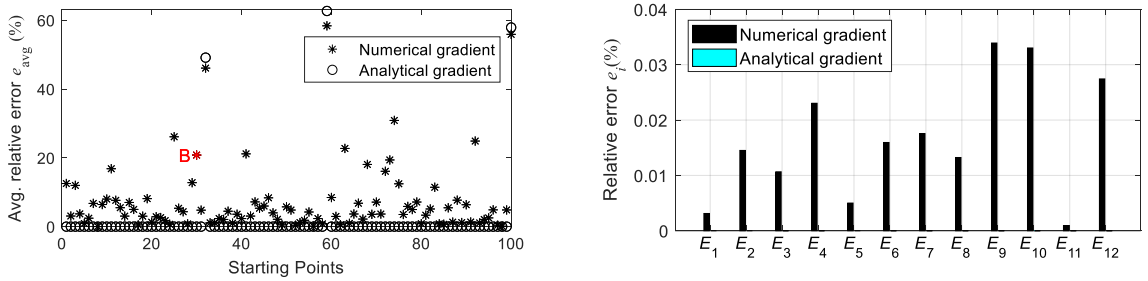


Figure 10. Hyperline walk between solution A to the global optimum

6.3 Model updating results: eigenvector difference approach

For the eigenvector difference formulation in Eq. (5), the residual vector length $n_{\mathbf{r}}$ equals $n_{\mathcal{M}} \cdot n_{\text{modes}} = 43 \cdot 3 = 129$. As a result, with the number of optimization variables $n_{\mathbf{x}}$ still equal to $n_{\boldsymbol{\alpha}} = 12$, the problem is determined and TRR algorithm is used. The numerical and analytical gradients are utilized for comparison. Figure 11(a) plots the average relative error e_{avg} of the stiffness updating variables for 100 random searches. Many starting points end with a large error using the numerical gradient while only three starting points converged with large errors using the analytical gradient. For each gradient case, the best solution is again selected as the one with the minimum objective function value among 100 result sets. The relative errors of the updated stiffness parameters for the best solution are plotted in Figure 11(b). Both the numerical and analytical gradient provide accurate updating results, and the analytical gradient provides no errors. To obtain the 100 optimal solutions, it took 22 hours 38 minutes using the numerical gradient while it took only 52 minutes using the analytical gradient.



(a) Average relative errors of updating variables (b) Relative errors of updated stiffness parameters

Figure 11. Model updating results for the concrete building frame by eigenvector difference approach using the TRR algorithm with numerical and analytical gradients

Similar to the MAC value approach, the nonconvexity of the eigenvector difference approach is studied using solution B in Figure 11(a). The average relative error of solution B is 20.81% and the objective function equals 0.0054. Figure 12 displays a walk from solution B to the global optimum along a hyperline in $\mathbb{R}^{12}$ space. Figure 12 clearly shows two valleys along the hyperline, indicating the non-convexity of the eigenvector difference approach in Eq. (5). Hence, global optimality cannot be ensured using off-the-shelf local optimization algorithms and multiple random starting points are necessary to obtain a better solution.

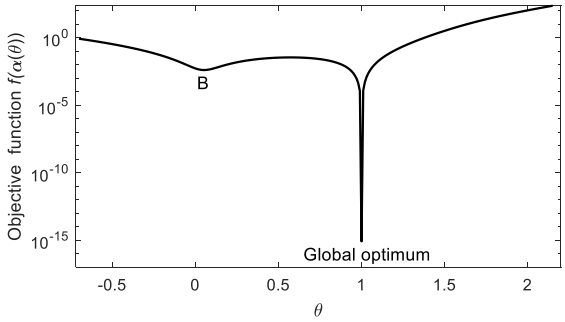


Figure 12. Hyperline walk between solution B to the global optimum

In summary, for model updating of the concrete building frame, the MAC value approach could not provide a set of optimal results with acceptable accuracy using either the numerical or analytical gradient during the optimization process. As for the eigenvector difference formulation, using the analytical gradient not only provides more accurate model updating results in general, but also can find the correct updating parameter values more efficiently. The study demonstrates the advantage of using the analytical gradient versus the numerical gradient.

7 Conclusions

An open-source MATLAB package for structural model updating called “SMU” is developed and shared with the research community. Current version 1.1 supports three model updating formulations in frequency domain, namely (i) the MAC (modal assurance criterion) value approach (Eq. (3)), (ii) the eigenvector difference approach (Eq. (5)), and (iii) the modal dynamic residual formulation (Eq. (6)). The first two belong to the family of modal property difference formulations. To facilitate the optimization process, analytical gradients of the formulations are derived and implemented in SMU. Considering that the objective functions of the model updating formulations are nonconvex, randomly generated starting points are adopted to increase the chance of finding the global minimum.

Among several structural examples provided in SMU, this paper demonstrates the SMU functionality using an 18-DOF model and a concrete building frame model in simulation. The MAC value approach and the modal dynamic residual formulation can correctly identify stiffness values for a relatively simpler structural model (the 18-DOF model), but may fail to provide reasonable results and increase computational cost when the complexity of a structure increases (as in the concrete building frame). The eigenvector difference approach can correctly update stiffness values for both simpler and more complex structures with better computational efficiency than the MAC value approach and the modal dynamic residual formulation. Furthermore, this study demonstrates the advantage of using the analytical gradient versus the numerical gradient. Using the analytical gradient during the optimization process in general not only provides more accurate model updating results, but also saves computational time, especially for a large-scale structure. For the concrete building frame, the computational time using the analytical gradient for the eigenvector difference approach is 52 minutes while it is 22 hours 38 minutes using the numerical gradient. For future work, different formulations and more structural examples are to be added. Moreover, the latest MATLAB 2021b version changed the Levenberg-Marquardt algorithm to support the bounds of optimization variables, and this change will be reflected in the SMU package in the future.

Funding

This research was partially funded by the National Science Foundation (CMMI-1150700 and CMMI-1634483). The first author received scholarship support from the Nakajima Foundation. Any opinions, findings, and conclusions expressed in this publication are those of the authors and do not necessarily reflect the view of the sponsors.

References

Bakir PG, Reynders E and De Roeck G (2008) An improved finite element model updating method by the global optimization technique 'Coupled Local Minimizers'. *Computers & Structures* 86(11-12): 1339-1352.

Boyd SP and Vandenberghe L (2004) *Convex Optimization.* Cambridge, UK: Cambridge University Press.

Branch MA, Coleman TF and Li Y (1999) A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing* 21(1): 1-23.

Byrd RH, Schnabel RB and Shultz GA (1988) Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical programming* 40(1-3): 247-263.

Chatzi EN and Smyth AW (2009) The unscented Kalman filter and particle filter methods for nonlinear structural system identification with non-collocated heterogeneous sensing. *Structural Control and Health Monitoring* 16(1): 99-123.

Coleman TF and Li Y (1996) An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization* 6(2): 418-445.

Dong X, Liu X, Wright T, et al. (2016) Validation of wireless sensing technology densely instrumented on a full-scale concrete frame structure. In: *Proceedings of International Conference on Smart Infrastructure and Construction (ICSIC)*, Cambridge, UK, pp.143-148.

Dong X and Wang Y (2018) Modal property difference formulations and optimization algorithm comparison towards FE model updating. In: *Proceedings of SPIE 2018, Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring* Denver, CO, USA, pp.1059828.

Dong X and Wang Y (2019) Finite element model updating of a steel pedestrian bridge model. In: *ASCE International Conference on Computing in Civil Engineering*, Atlanta, GA, USA, pp.397-404.

Doucet A and Tadić VB (2003) Parameter estimation in general state-space models using particle methods. *Annals of the institute of Statistical Mathematics* 55(2): 409-422.

Farhat C and Hemez FM (1993) Updating finite element dynamic models using an element-by-element sensitivity methodology. *AIAA Journal* 31(9): 1702-1711.

Fox R and Kapoor M (1968) Rates of change of eigenvalues and eigenvectors. *AIAA journal* 6(12): 2426-2429.

Friswell MI and Mottershead JE (1995) *Finite element model updating in structural dynamics.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Hofmeister B, Bruns M and Rolfes R (2019) Finite element model updating using deterministic optimisation: A global pattern search approach. *Engineering Structures* 195: 373-381.

Kane M, Zhu D, Hirose M, et al. (2014) Development of an extensible dual-core wireless sensing node for cyber-physical systems. In: *Proceedings of SPIE 2014, Nondestructive Characterization for Composite Materials, Aerospace Engineering, Civil Infrastructure, and Homeland Security*, San Diego, CA, USA, pp.90611U.

Kosmatka JB and Ricles JM (1999) Damage detection in structures by modal vibration characterization. *ASCE Journal of Structural Engineering* 125(12): 1384-1392.

LeVeque RJ (2007) *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems.* Philadelphia, PA, USA: SIAM.

Li D, Dong X and Wang Y (2018) Model updating using sum of squares (SOS) optimization to minimize modal dynamic residuals. *Structural Control and Health Monitoring* 25(12): e2263.

Lynch JP and Loh KJ (2006) A summary review of wireless sensors and sensor networks for structural health monitoring. *The Shock and Vibration Digest* 38(2): 91-128.

MathWorks Inc. (2019) *Optimization Toolbox™ User's Guide.* Natick, MA, USA: MathWorks Inc.

Moller PW and Friberg O (1998) Updating large finite element models in structural dynamics. *AIAA journal* 36(10): 1861-1868.

Moré J (1978) The Levenberg-Marquardt algorithm: Implementation and theory. *Numerical Analysis*. Berlin, Germany: Springer pp.105-116.

Nelson RB (1976) Simplified calculation of eigenvector derivatives. *AIAA journal* 14(9): 1201-1205.

Otsuki Y, Li D, Dey SS, et al. (2021a) Finite element model updating of an 18-story structure using branch-and-bound algorithm with epsilon-constraint. *Journal of Civil Structural Health Monitoring*. doi: 10.1007/s13349-020-00468-3 (in print).

Otsuki Y, Li D, Xinjun D, et al. (2021b) SMU – an open-source MATLAB package for structural model updating. In: *Proceedings of the 10th International Conference on Bridge Maintenance, Safety and Management (IABMAS)*, Sapporo, Japan, pp.1621-1628.

Salawu OS (1997) Detection of structural damage through changes in frequency: A review. *Engineering Structures* 19(9): 718-723.

Sato T and Qi K (1998) Adaptive H∞ filter: its application to structural identification. *Journal of Engineering Mechanics* 124(11): 1233-1240.

Suita K, Suzuki Y and Takahashi M (2015) Collapse behavior of an 18-story steel moment frame during a shaking table test. *International Journal of High-Rise Buildings* 4(3): 171-180.

Teughels A, Maeck J and De Roeck G (2002) Damage assessment by FE model updating using damage functions. *Computers & Structures* 80(25): 1869-1879.

Wang Y, Dong X and Li D (2019a) A non-convexity study in finite element model updating. In: *Proceedings of the 9th International Conference on Structural Health Monitoring of Intelligent Infrastructure (SHMII-9)*, St. Louis, MO, USA, pp.24-29.

Wang Y, Dong X, Li D, et al. (2019b) SMU: MATLAB Package for Structural Model Updating, version 1.1. Available at: https://github.com/ywang-structures/Structural-Model-Updating.

Weng S, Xia Y, Xu Y-L, et al. (2011) Substructure based approach to finite element model updating. *Computers & Structures* 89(9-10): 772-782.

Yang JN, Lin SL, Huangl HW, et al. (2006) An adaptive extended Kalman filter for structural damage identification. *Structural Control & Health Monitoring* 13(4): 849-867.

Yuen KV (2012) Updating large models for mechanical systems using incomplete modal measurement. *Mechanical Systems and Signal Processing* 28: 297-308.

Zhang QW, Chang CC and Chang TYP (2000) Finite element model updating for structures with parametric constraints. *Earthquake Engineering & Structural Dynamics* 29(7): 927-944.

Zhang Z and Sun C (2020) Structural damage identification via physics-guided machine learning: a methodology integrating pattern recognition with finite element model updating. *Structural Health Monitoring*. doi: 10.1177/1475921720927488 (in print).

Zhu D, Dong X and Wang Y (2016) Substructure stiffness and mass updating through minimization of modal dynamic residuals. *ASCE Journal of Engineering Mechanics* 142(5): 04016013.

Zhu H, Li J, Tian W, et al. (2021) An enhanced substructure-based response sensitivity method for finite element model updating of large-scale structures. *Mechanical Systems and Signal Processing* 154: 107359.